EMNLP 2008

2008 Conference on Empirical Methods in Natural Language Processing

Proceedings of the Conference

25–27 October 2008
Honolulu, Hawaii, USA

A meeting of SIGDAT, a Special Interest Group of the ACL

Production and Manufacturing by
Omnipress Inc.
2600 Anderson Street
Madison, WI 53707
USA

Preface

Welcome to the 2008 Conference on Empirical Methods in Natural Language Processing! The conference is organized under the auspices of SIGDAT, the ACL Special Interest Group for linguistic data and corpus-based approaches to natural language processing. It is co-located this year with AMTA 2008 and the International Workshop on Spoken Language Translation, in Honolulu, Hawaii.

EMNLP received 385 submissions. We were able to accept 116 papers in total (an acceptance rate of 30%). 81 of the papers (21%) were accepted for oral presentation, and 35 (9%) for poster presentation. Two poster papers were subsequently withdrawn after acceptance. The papers were selected by a program committee of 15 area chairs, from Asia, Europe, and North America, assisted by a panel of 339 reviewers. This year EMNLP introduced an author response period. Authors were able to read and respond to the reviews of their paper before the program committee made a final decision. They were asked to correct factual errors in the reviews and answer questions raised in the reviewer comments. The intention was to help produce more accurate reviews. In some cases, reviewers changed their scores in view of the authors' response and the area chairs read all responses carefully prior to making recommendations for acceptance.

First and foremost, we would like to thank the authors who submitted their work to EMNLP. The sheer number of submissions reflects how broad and active our field is. We are deeply indebted to the area chairs and the reviewers for their hard work. They enabled us to select an exciting program and to provide valuable feedback to the authors. We are grateful to our invited speakers Oren Etzioni, Tom Griffiths, and Fernando Pereira who graciously agreed to give talks at EMNLP. Additional thanks to the Publications Chair, Sebastian Padó, who put this volume together. Jason Eisner helped us immensely by compiling a web site on "How to Serve as Program Chair of a Conference" (`http://www.cs.jhu.edu/~jason/advice/how-to-chair-a-conference.html`). Special thanks to David Yarowsky and Ken Church of SIGDAT who provided much valuable advice and assistance over the past months. David also helped raise important financial support for the conference. We are most grateful to Priscilla Rasmussen who helped us with various logistic and organizational aspects of the conference. Rich Gerber and the START team responded to our questions quickly, and helped us manage the large number of submissions smoothly. Finally, thanks are due to our webmaster, Francesco Figari, who revamped our conference website on very short notice.

We hope you enjoy the conference!

Mirella Lapata and Hwee Tou Ng
EMNLP 2008 Program Co-Chairs

Reviewers:

Meni Adler, Eugene Agichtein, Amr Ahmed, Yaser Al-Onaizan, Galen Andrew;

Peter Bailey, Timothy Baldwin, Carmen Banea, Roy Bar-Haim, Regina Barzilay, Roberto Basili, Cosmin Adrian Bejan, Anja Belz, Daniel Bikel, Misha Bilenko, Alexandra Birch, Alan Black, Elizabeth Boschee, Jordan Boyd-Graber, Eric Breck, Peter Bruza, Ivan Bulyko, Razvan Bunescu, Harry Bunt, John Burger, Bill Byrne, Donna Byron;

Chris Callison-Burch, Nicoletta Calzolari, Claire Cardie, Xavier Carreras, Vittorio Castelli, Neus Català, Dan Cer, Özgür Çetin, Nate Chambers, Pi-Chuan Chang, Eugene Charniak, Ciprian Chelba, Hsin-Hsi Chen, Colin Cherry, David Chiang, Yejin Choi, Jennifer Chu-Carroll, Tat-Seng Chua, Ken Church, Massimiliano Ciaramita, Alexander Clark, Stephen Clark, James Clarke, Michael Collins, Koby Crammer, Mathias Creutz, Dan Cristea, Silviu Cucerzan, Hang Cui;

Jan Daciuk, Walter Daelemans, Sajib Dasgupta, Hal Daume, Marie-Catherine de Marneffe, Maarten de Rijke, Steve DeNeefe, John DeNero, Mona Diab, Shilin Ding, Bill Dolan, Mark Dras, Mark Dredze, Kevin Duh, Chris Dyer, Myroslava Dzikovska;

Phil Edmonds, Jason Eisner, Michael Elhadad, Katrin Erk;

Hui Fang, Marcello Federico, Raquel Fernandez, Jenny Finkel, Alex Fraser, Marjorie Freedman, Dayne Freitag, Atsushi Fujii, Pascale Fung;

Ryan Gabbard, Robert Gaizauskas, Michel Galley, Michael Gamon, Kuzman Ganchev, Jianfeng Gao, Ruifang Ge, Darren Gergle, Ulrich Germann, Dan Gildea, Jonathan Ginzburg, Roxana Girju, Amir Globerson, Sharon Goldwater, Joao Graca, Ralph Grishman;

Nizar Habash, Aria Haghighi, Udo Hahn, Dilek Hakkani-Tur, Keith Hall, Jirka Hana, Sanda Harabagiu, Mary Harper, Mark Hasegawa-Johnson, Timothy Hazen, Xiaodong He, James Henderson, John Henderson, Ulf Hermjakob, Andrew Hickl, Ryuichiro Higashinaka, Dustin Hillard, Graeme Hirst, Julia Hockenmaier, Beth Ann Hockey, Véronique Hoste, Wu Hua, Jimmy Huang, Liang Huang;

Nancy Ide, Diana Inkpen, Hideki Isozaki, Abe Ittycheriah;

Heng Ji, Jing Jiang, Valentin Jijkoun, Howard Johnson, Mark Johnson, Rie Johnson, Kristiina Jokinen, Pamela Jordan, Joemon Jose, Aravind Joshi;

Michael Kaisser, Min-Yen Kan, Hiroshi Kanayama, Noriko Kando, Nikiforos Karamanis, Rohit Kate, Junichi Kazama, Frank Keller, Sanjeev Khudanpur, Katrin Kirchhoff, Kevin Knight, Philipp Koehn, Alexander Koller, Terry Koo, Moshe Koppel, Valia Kordoni, Anna Korhonen, Taku Kudo, Sandra Kuebler, Roland Kuhn, Alex Kulesza, Ravi Kumar, Shankar Kumar;

Simon Lacoste-Julien, Wai Lam, Philippe Langlais, Guy Lapalme, Guy Lebanon, Gary Geunbae Lee, Lillian Lee, Gina Levow, Roger Levy, Chi-Ho Li, Xiao Li, Zhifei Li, Percy Liang, Ee-Peng Lim, Kenneth Litkowski, Bing Liu, Yang Liu, Yang Liu, Karen Livescu, Jose Luis Vicedo, Xiaoqiang Luo, Yajuan Lv;

Bill MacCartney, Bernardo Magnini, Gideon Mann, Daniel Marcu, Katja Markert, David Martinez, Arne Mauser, Diana McCarthy, David McClosky, David McDonald, Ryan McDonald, Kathy McKeown, Michael McTear, Arul Menezes, Donald Metzler, Rada Mihalcea, Gilad Mishne, Teruko Mitamura, Diego Molla Aliod, Christof Monz, Robert Moore, Alessandro Moschitti;

Tetsuji Nakagawa, Satoshi Nakamura, Preslav Nakov, Vivi Nastase, Roberto Navigli, Ani Nenkova, Vincent Ng, Grace Ngai, Patrick Nguyen, Gabriel Nicolae;

Franz Och, Kemal Oflazer, Arzucan Özgür;

Sebastian Padó, Tim Paek, Martha Palmer, Patrick Pantel, Marius Pasca, Marco Pennacchiotti, Slav Petrov, Joseph Picone, Ana-Maria Popescu, Victor Poznanski, Sameer Pradhan, John Prager, Kathrin Probst, Vasin Punyakanok;

Tao Qin;

Dan Ramage, Owen Rambow, Deepak Ravichandran, Norbert Reithinger, Philip Resnik, German Rigau, Hae-Chang Rim, Fabio Rinaldi, Brian Roark, Patrick Ruch;

Yoshinori Sagisaka, Magnus Sahlgren, Tetsuya Sakai, David Schlangen, Helmut Schmid, Holger Schwenk, Frédérique Segond, Yohei Seki, Satoshi Sekine, Stephanie Seneff, Izhak Shafran, Libin Shen, Takahiro Shinozaki, Advaith Siddharthan, Khalil Sima'an, Manhung Siu, David Smith, Noah Smith, Rion Snow, Dawei Song, Mark Steedman, Amanda Stent, Mark Stevenson, Veselin Stoyanov, Michael Strube, Le Sun, Maosong Sun, Mihai Surdeanu, Charles Sutton, Stan Szpakowicz, Idan Szpektor;

Hiroya Takamura, Partha Pratim Talukdar, Marta Tatu, Simone Teufel, Christoph Tillmann, Ivan Titov, David Traum, Reut Tsarfaty, Huihsin Tseng, Junichi Tsujii, Dan Tufiş, Gökhan Tür, Joseph Turian, Evelyne Tzoukermann;

Kiyotaka Uchimoto, Takehito Utsuro;

Lucy Vanderwende, Stephan Vogel, Ellen Voorhees, Piek Vossen;

Stephen Wan, Qin Wang, Wei Wang, Wen Wang, Ye-Yi Wang, Taro Watanabe, Bonnie Webber, David Weir, Ralph Weischedel, Michael White, Richard Wicentowski, Dominic Widdows, Jason Williams, Theresa Wilson, Shuly Wintner, Yuk Wah Wong, Britta Wrede, Dekai Wu;

Fei Xia, Jinxi Xu, Jun Xu, Peng Xu, Guirong Xue;

Mei Yang, Qiang Yang, Scott Yih, Deniz Yuret;

Annie Zaenen, David Zajic, Richard Zens, Luke Zettlemoyer, Dongdong Zhang, Hao Zhang, Tong Zhang, Bing Zhao, Jing Zheng, GuoDong Zhou, Joe Zhou, Jerry Zhu, Imed Zitouni, Onno Zoeter, Andreas Zollmann, Ingrid Zukerman, Geoff Zweig

Table of Contents

Conference Program Overview

Saturday, October 25, 2008

9:15–10:30	Session 1: Plenary Session
10:30–11:00	Morning Break
11:00–12:15	Sessions 2a, 2b and 2c
12:15–14:00	Lunch
14:00–15:15	Sessions 3a, 3b and 3c
15:15–15:45	Afternoon Break
15:45–17:00	Sessions 4a, 4b and 4c
18:00–21:00	Session 5: All Posters

Sunday, October 26, 2008

9:30–10:30	Session 6: Plenary Session
10:30–11:00	Morning Break
11:00–12:15	Sessions 7a, 7b and 7c
12:15–14:00	Lunch
14:00–15:15	Sessions 8a, 8b and 8c
15:15–15:45	Afternoon Break
15:45–17:00	Sessions 9a, 9b and 9c

Monday, October 27, 2008

9:30–10:30	Session 10: Plenary Session
10:30–11:00	Morning Break
11:00–12:15	Sessions 11a, 11b and 11c
12:15-13:00	SIGDAT Business Meeting
13:00–14:00	Lunch
14:00–15:15	Sessions 12a, 12b and 12c
15:15–15:45	Afternoon Break
15:45–17:00	Sessions 13a, 13b and 13c

Conference Program

Saturday, October 25, 2008

	Session 1: Plenary Session
9:15–9:30	Opening Remarks
9:30–10:30	Invited Talk: *We KnowItAll: Lessons from a Quarter Century of Web Extraction Research* Oren Etzioni, University of Washington
10:30-11:00	**Morning Break**
	Session 2a: Information Extraction
11:00–11:25	*Revealing the Structure of Medical Dictations with Conditional Random Fields* Jeremy Jancsary, Johannes Matiasek and Harald Trost
11:25–11:50	*It's a Contradiction – no, it's not: A Case Study using Functional Relations* Alan Ritter, Stephen Soderland, Doug Downey and Oren Etzioni
11:50–12:15	*Regular Expression Learning for Information Extraction* Yunyao Li, Rajasekar Krishnamurthy, Sriram Raghavan, Shivakumar Vaithyanathan and H. V. Jagadish
	Session 2b: Machine Learning
11:00–11:25	*Modeling Annotators: A Generative Approach to Learning from Annotator Rationales* Omar Zaidan and Jason Eisner
11:25–11:50	*One-Class Clustering in the Text Domain* Ron Bekkerman and Koby Crammer
11:50–12:15	*Refining Generative Language Models using Discriminative Learning* Ben Sandbank

Saturday, October 25, 2008 (continued)

Saturday, October 25, 2008 (continued)

Saturday, October 25, 2008 (continued)

Monday, October 27, 2008 (continued)

Revealing the Structure of Medical Dictations with Conditional Random Fields

Jeremy Jancsary and **Johannes Matiasek**
Austrian Research Institute for Artificial Intelligence
A-1010 Vienna, Freyung 6/6
firstname.lastname@ofai.at

Harald Trost
Department of Medical Cybernetics and Artificial Intelligence
of the Center for Brain Research, Medical University Vienna, Austria
harald.trost@meduniwien.ac.at

Abstract

Automatic processing of medical dictations poses a significant challenge. We approach the problem by introducing a statistical framework capable of identifying types and boundaries of sections, lists and other structures occurring in a dictation, thereby gaining explicit knowledge about the function of such elements. Training data is created semi-automatically by aligning a parallel corpus of corrected medical reports and corresponding transcripts generated via automatic speech recognition. We highlight the properties of our statistical framework, which is based on conditional random fields (CRFs) and implemented as an efficient, publicly available toolkit. Finally, we show that our approach is effective both under ideal conditions and for real-life dictation involving speech recognition errors and speech-related phenomena such as hesitation and repetitions.

1 Introduction

It is quite common to dictate reports and leave the typing to typists – especially for the medical domain, where every consultation or treatment has to be documented. Automatic Speech Recognition (ASR) can support professional typists in their work by providing a transcript of what has been dictated. However, manual corrections are still needed. In particular, speech recognition errors have to be corrected. Furthermore, speaker errors, such as hesitations or repetitions, and instructions to the transcriptionist have to be removed. Finally, and most notably, proper structuring and formatting of the report has to be performed. For the medical domain, fairly clear guidelines exist with regard to what has to be dictated, and how it should be arranged. Thus, missing headings may have to be inserted, sentences must be grouped into paragraphs in a meaningful way, enumeration lists may have to be introduced, and so on.

The goal of the work presented here was to ease the job of the typist by formatting the dictation according to its structure and the formatting guidelines. The prerequisite for this task is the identification of the various structural elements in the dictation which will be be described in this paper.

```
complaint dehydration weakness and diarrhea
full stop Mr.  Will Shawn is a 81-year-old
cold Asian gentleman who came in with fever
and Persian diaper was sent to the emergency
department by his primary care physician due
him being dehydrated period ... neck physical
exam general alert and oriented times three
known acute distress vital signs are stable
... diagnosis is one chronic diarrhea with
hydration he also has hypokalemia neck number
thromboctopenia probably duty liver cirrhosis
... a plan was discussed with patient in
detail will transfer him to a nurse and
facility for further care ... end of dictation
```

Fig. 1: Raw output of speech recognition

Figure 1 shows a fragment of a typical report as recognized by ASR, exemplifying some of the problems we have to deal with:

- Punctuation and enumeration markers may be dictated or not, thus sentence boundaries and numbered items often have to be inferred;
- the same holds for (sub)section headings;
- finally, recognition errors complicate the task.

```
CHIEF COMPLAINT
Dehydration, weakness and diarrhea.

HISTORY OF PRESENT ILLNESS
Mr.  Wilson is a 81-year-old Caucasian
gentleman who came in here with fever and
persistent diarrhea.  He was sent to the
emergency department by his primary care
physician due to him being dehydrated.
...

PHYSICAL EXAMINATION
  GENERAL: He is alert and oriented times
     three, not in acute distress.

  VITAL SIGNS: Stable.
...

DIAGNOSIS
  1. Chronic diarrhea with dehydration.  He
     also has hypokalemia.

  2. Thromboctopenia, probably due to liver
     cirrhosis.
...

PLAN AND DISCUSSION
The plan was discussed with the patient
in detail.  Will transfer him to a nursing
facility for further care.
...
```

Fig. 2: A typical medical report

When properly edited and formatted, the same dictation appears significantly more comprehensible, as can be seen in figure 2. In order to arrive at this result it is necessary to identify the inherent structure of the dictation, i.e. the various hierarchically nested segments. We will recast the segmentation problem as a multi-tiered tagging problem and show that indeed a good deal of the structure of medical dictations can be revealed.

The main contributions of our paper are as follows: First, we introduce a generic approach that can be integrated seamlessly with existing ASR solutions and provides structured output for medical dictations. Second, we provide a freely available toolkit for factorial conditional random fields (CRFs) that forms the basis of aforementioned approach and is also applicable to numerous other problems (see section 6).

2 Related Work

The structure recognition problem dealt with here is closely related to the field of *linear text segmentation* with the goal to partition text into coherent blocks, but on a single level. Thus, our task generalizes linear text segmentation to multiple levels.

A meanwhile classic approach towards domain-independent linear text segmentation, *C99*, is presented in Choi (2000). *C99* is the baseline which many current algorithms are compared to. Choi's algorithm surpasses previous work by Hearst (1997), who proposed the *Texttiling* algorithm. The best results published to date are – to the best of our knowledge – those of Lamprier et al. (2008).

The automatic detection of (sub)section *topics* plays an important role in our work, since changes of topic indicate a section boundary and appropriate headings can be derived from the section type. Topic detection is usually performed using methods similar to those of *text classification* (see Sebastiani (2002) for a survey).

Matsuov (2003) presents a dynamic programming algorithm capable of segmenting medical reports into sections and assigning topics to them. Thus, the aims of his work are similar to ours. However, he is not concerned with the more fine-grained elements, and also uses a different machinery.

When dealing with tagging problems, statistical frameworks such as HMMs (Rabiner, 1989) or, recently, CRFs (Lafferty et al., 2001) are most commonly applied. Whereas HMMs are generative models, CRFs are discriminative models that can incorporate rich features. However, other approaches to text segmentation have also been pursued. E.g., McDonald et al. (2005) present a model based on multilabel classification, allowing for natural handling of overlapping or non-contiguous segments.

Finally, the work of Ye and Viola (2004) bears similarities to ours. They apply CRFs to the parsing of hierarchical lists and outlines in handwritten notes, and thus have the same goal of finding deep structure using the same probabilistic framework.

3 Problem Representation

For representing our segmentation problem we use a trick that is well-known from chunking and named entity recognition, and recast the problem as a tagging problem in the so-called BIO[1] notation. Since we want to assign a type to every segment, OUTSIDE labels are not needed. However, we perform seg-

[1] BEGIN - INSIDE - OUTSIDE

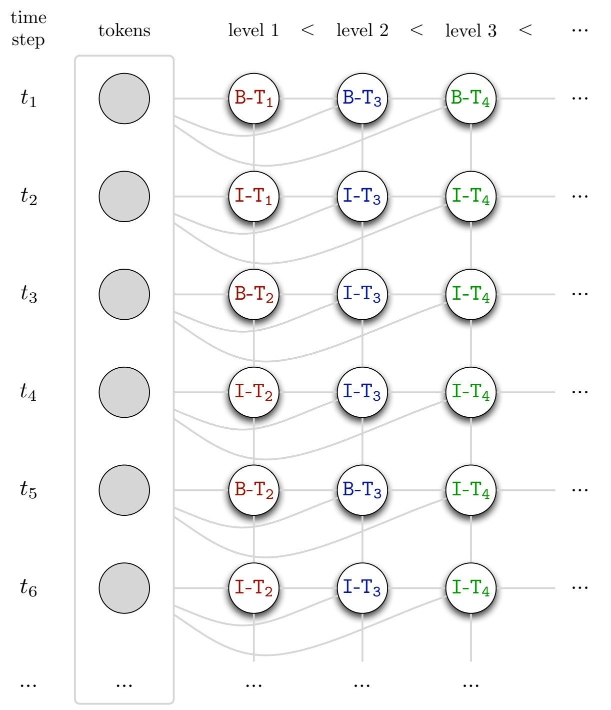


Fig. 3: Multi-level segmentation as tagging problem

mentation on multiple levels, therefore multiple *label chains* are required. Furthermore, we also want to assign *types* to certain segments, thus the labels need an encoding for the type of segment they represent. Figure 3 illustrates this representation: B-T_i denotes the beginning of a segment of type T_i, while I-T_i indicates that the segment of type T_i continues. By adding label chains, it is possible to group the segments of the previous chain into coarser units. Tree-like structures of unlimited depth can be expressed this way[2]. The gray lines in figure 3 denote dependencies between nodes. Node labels also depend on the input token sequence in an arbitrarily wide context window.

4 Data Preparation

The raw data available to us consists of two parallel corpora of 2007 reports from the area of medical consultations, dictated by physicians. The first corpus, C_{RCG}, consists of the raw output of ASR (figure 1), the other one, C_{COR}, contains the corresponding corrected and formatted reports (figure 2).

In order to arrive at an annotated corpus in a format suitable for the tagging problem, we first have to analyze the report structure and define appropriate labels for each segmentation level. Then, every token has to be annotated with the appropriate begin or inside labels. A report has 625 tokens on average, so the manual annotation of roughly 1.25 million tokens seemed not to be feasible. Thus we decided to produce the annotations programmatically and restrict manual work to corrections.

4.1 Analysis of report structure

When inspecting reports in C_{COR}, a human reader can easily identify the various elements a report consists of, such as *headings* – written in bold on a separate line – introducing sections, *subheadings* – written in bold followed by a colon – introducing subsections, and *enumerations* starting with indented numbers followed by a period. Going down further, there are *paragraphs* divided into *sentences*. Using these structuring elements, a hierarchic data structure comprising all report elements can be induced.

Sections and subsections are typed according to their heading. There exist clear recommendations on structuring medical reports, such as E2184-02 (ASTM International, 2002). However, actual medical reports still vary greatly with regard to their structure. Using the aforementioned standard, we assigned the (sub)headings that actually appeared in the data to the closest type, introducing new types only when absolutely necessary. Finally we arrived at a structure model with three label chains:

- **Sentence level**, with 4 labels: `Heading`, `Subheading`, `Sentence`, `Enummarker`
- **Subsection level**, with 45 labels: `Paragraph`, `Enumelement`, `None` and 42 subsection types (e.g. `VitalSigns`, `Cardiovascular` ...)
- **Section level**, with 23 section types (e.g. `ReasonForEncounter`, `Findings`, `Plan` ...)

4.2 Corpus annotation

Since the reports in C_{COR} are manually edited they are reliable to parse. We employed a broad-coverage dictionary (handling also multi-word terms) and a domain-specific grammar for parsing and layout information. A regular heading grammar was used for mapping (sub)headings to the defined (sub)section labels (for details see Jancsary (2008)). The output

[2]Note, that since we omit a redundant top-level chain, this structure technically is a *hedge* rather than a *tree*.

C_{COR}		OP	C_{RCG}	
...	...	...	...	...
B − Head	CHIEF	del		
Head	COMPLAINT	sub	complaint	B − Head
B − Sent	Dehydration	sub	dehydration	B − Sent
Sent	,	del		
Sent	weakness	sub	weakness	Sent
Sent	and	sub	and	Sent
Sent	diarrhea	sub	diarrhea	Sent
Sent	.	sub	fullstop	Sent
B − Sent	Mr.	sub	Mr.	B − Sent
Sent	Wilson	sub	Will	Sent
		ins	Shawn	Sent
Sent	is	sub	is	Sent
Sent	a	sub	a	Sent
Sent	81-year-old	sub	81-year-old	Sent
Sent	Caucasian	sub	cold	Sent
Sent		ins	Asian	Sent
Sent	gentleman	sub	gentleman	Sent
Sent	who	sub	who	Sent
Sent	came	sub	came	Sent
Sent	in	del		
Sent	here	sub	here	Sent
Sent	with	sub	with	Sent
Sent	fever	sub	fever	Sent
Sent	and	sub	and	Sent
Sent	persistent	sub	Persian	Sent
Sent	diarrhea	sub	diaper	Sent
Sent	.	del		
...	...	...	...	...

Fig. 4: Mapping labels via alignment

of the parser is a hedge data structure from which the annotation labels can be derived easily.

However, our goal is to develop a model for recognizing the report structure from the *dictation*, thus we have to map the newly created annotation of reports in C_{COR} onto the corresponding reports in C_{RCG}. The basic idea here is to align the tokens of C_{COR} with the tokens in C_{RCG} and to copy the annotations (cf. figure 4[3]). There are some peculiarities we have to take care of during alignment:

1. non-dictated items in C_{COR} (e.g. punctuation, headings)
2. dictated words that do not occur in C_{COR} (meta instructions, repetitions)
3. non-identical but corresponding items (recognition errors, reformulations)

Since it is particularly necessary to correctly align items of the third group, standard string-edit distance based methods (Levenshtein, 1966) need to be augmented. Therefore we use a more sophisticated cost function. It assigns tokens that are similar (either from a semantic or phonetic point of view) a low cost for substitution, whereas dissimilar tokens receive a prohibitively expensive score. Costs for deletion and insertion are assigned inversely. Semantic similarity is computed using Wordnet (Fellbaum, 1998) and UMLS (Lindberg et al., 1993). For phonetic matching, the Metaphone algorithm (Philips, 1990) was used (for details see Huber et al. (2006)).

4.3 Feature Generation

The annotation discussed above is the first step towards building a training corpus for a CRF-based approach. What remains to be done is to provide observations for each time step of the observed entity, i.e. for each token of a report; these are expected to give hints with regard to the annotation labels that are to be assigned to the time step. The observations, associated with one or more annotation labels, are usually called *features* in the machine learning literature. During CRF training, the parameters of these features are determined such that they indicate the significance of the observations for a certain label or label combination; this is the basis for later tagging of unseen reports.

We use the following features for each time step of the reports in C_{COR} and C_{RCG}:

- **Lexical features** covering the local context of ± 2 tokens (e.g., `patient@0`, `the@-1`, `is@1`)
- **Syntactic features** indicating the possible syntactic categories of the tokens (e.g., `NN@0`, `JJ@0`, `DT@-1` and `be+VBZ+aux@1`)
- **Bag-of-word (BOW) features** intend to capture the topic of a text segment in a wider context of ± 10 tokens, without encoding any order. Tokens are lemmatized and replaced by their UMLS *concept IDs*, if available, and weighed by `TF`. Thus, different words describing the same concept are considered equal.
- **Semantic type features** as above, but using UMLS *semantic types* instead of concept IDs provide a coarser level of description.
- **Relative position features**: The report is divided into eight parts corresponding to eight binary features; only the feature corresponding to the part of the current time step is set.

[3]This approach can easily be generalized to multiple label chains.

5 Structure Recognition with CRFs

Conditional random fields (Lafferty et al., 2001) are conditional models in the exponential family. They can be considered a generalization of multinomial logistic regression to output with non-trivial internal structure, such as sequences, trees or other graphical models. We loosely follow the general notation of Sutton and McCallum (2007) in our presentation.

Assuming an undirected graphical model G over an observed entity $\boldsymbol{x}$ and a set of discrete, interdependent random variables[4] $\boldsymbol{y}$, a conditional random field describes the conditional distribution:

$$p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{x})} \prod_{c \in G} \boldsymbol{\phi}_c(\boldsymbol{y}_c, \boldsymbol{x}; \boldsymbol{\theta}_c) \qquad (1)$$

The normalization term $Z(\boldsymbol{x})$ sums over all possible joint outcomes of $\boldsymbol{y}$, i.e.,

$$Z(\boldsymbol{x}) = \sum_{\boldsymbol{y}'} p(\boldsymbol{y}'|\boldsymbol{x};\boldsymbol{\theta}) \qquad (2)$$

and ensures the probabilistic interpretation of $p(\boldsymbol{y}|\boldsymbol{x})$. The graphical model G describes interdependencies between the variables $\boldsymbol{y}$; we can then model $p(\boldsymbol{y}|\boldsymbol{x})$ via factors $\boldsymbol{\phi}_c(\cdot)$ that are defined over cliques $c \in G$. The factors $\boldsymbol{\phi}_c(\cdot)$ are computed from sufficient statistics $\{f_{ck}(\cdot)\}$ of the distribution (corresponding to the features mentioned in the previous section) and depend on possibly overlapping sets of parameters $\boldsymbol{\theta}_c \subseteq \boldsymbol{\theta}$ which together form the parameters $\boldsymbol{\theta}$ of the conditional distribution:

$$\boldsymbol{\phi}_c(\boldsymbol{y}_c, \boldsymbol{x}; \boldsymbol{\theta_c}) = \exp\left(\sum_{k=1}^{|\boldsymbol{\theta}_c|} \lambda_{ck} f_{ck}(\boldsymbol{x}, \boldsymbol{y}_c)\right) \qquad (3)$$

In practice, for efficiency reasons, independence assumptions have to be made about variables $y \in \boldsymbol{y}$, so G is restricted to small cliques (say, $(|c| \leq 3)$. Thus, the sufficient statistics only depend on a limited number of variables $\boldsymbol{y_c} \subseteq \boldsymbol{y}$; they can, however, access the whole observed entity $\boldsymbol{x}$. This is in contrast to generative approaches which model a joint distribution $p(\boldsymbol{x}, \boldsymbol{y})$ and therefore have to extend the independence assumptions to elements $x \in \boldsymbol{x}$.

[4]In our case, the discrete outcomes of the random variables $\boldsymbol{y}$ correspond to the annotation labels described in the previous section.

The factor-specific parameters $\boldsymbol{\theta}_c$ of a CRF are typically tied for certain cliques, according to the problem structure (i.e., $\boldsymbol{\theta}_{c_1} = \boldsymbol{\theta}_{c_2}$ for two cliques c_1, c_2 with tied parameters). E.g., parameters are usually tied across time if G is a sequence. The factors can then be partitioned into a set of *clique templates* $\mathcal{C} = \{C_1, C_2, \ldots C_P\}$, where each clique template C_p is a set of factors with tied parameters $\boldsymbol{\theta}_p$ and corresponding sufficient statistics $\{f_{pk}(\cdot)\}$. The CRF can thus be rewritten as:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \prod_{C_p \in \mathcal{C}} \prod_{\boldsymbol{\phi}_c \in C_p} \boldsymbol{\phi}_c(\boldsymbol{y}_c, \boldsymbol{x}; \boldsymbol{\theta}_p) \qquad (4)$$

Furthermore, in practice, the sufficient statistics $\{f_{pk}(\cdot)\}$ are computed from a subset $\boldsymbol{x}_c \subseteq \boldsymbol{x}$ that is relevant to a factor $\boldsymbol{\phi}_c(\cdot)$. In a sequence labelling task, tokens $x \in \boldsymbol{x}$ that are in temporal proximity to an output variable $y \in \boldsymbol{y}$ are typically most useful. Nevertheless, in our notation, we will let factors depend on the whole observed entity $\boldsymbol{x}$ to denote that all of $\boldsymbol{x}$ can be accessed if necessary.

For our structure recognition task, the graphical model G exhibits the structure shown in figure 3, i.e., there are multiple connected chains of variables with factors defined over single-node cliques and two-node cliques within and between chains; the parameters of factors are tied across time. This corresponds to the *factorial CRF* structure described in Sutton and McCallum (2005). Structure recognition using conditional random fields then involves two separate steps: *parameter estimation*, or training, is concerned with selecting the parameters of a CRF such that they fit the given training data. *Prediction*, or testing, determines the best label assignment for unknown examples.

5.1 Parameter estimation

Given IID training data $\mathcal{D} = \{\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\}_{i=1}^{N}$, parameter estimation determines:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}'}{\operatorname{argmax}} \left(\sum_{i}^{N} p(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)}; \boldsymbol{\theta}')\right) \qquad (5)$$

i.e., those parameters that maximize the conditional probability of the CRF given the training data.

In the following, we will not explicitly sum over $_{i=1}^{N}$; as Sutton and McCallum (2007) note, the training instances $\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}$ can be considered disconnected components of a single undirected model G.

We thus assume G and its factors $\boldsymbol{\phi}_c(\cdot)$ to extend over all training instances. Unfortunately, (5) cannot be solved analytically. Typically, one performs maximum likelihood estimation (MLE) by maximizing the conditional log-likelihood numerically:

$$\ell(\boldsymbol{\theta}) = \sum_{C_p \in \mathcal{C}} \sum_{\boldsymbol{\phi}_c \in C_p} \sum_{k=1}^{|\boldsymbol{\theta}_p|} \lambda_{pk} f_{pk}(\boldsymbol{x}, \boldsymbol{y}_c) - \log Z(\boldsymbol{x}) \tag{6}$$

Currently, limited-memory gradient-based methods such as LBFGS (Nocedal, 1980) are most commonly employed for that purpose[5]. These require the partial derivatives of (6), which are given by:

$$\frac{\partial \ell}{\partial \lambda_{pk}} = \sum_{\boldsymbol{\phi}_c \in C_p} f_{pk}(\boldsymbol{x}, \boldsymbol{y}_c) - \sum_{\boldsymbol{y}'_c} f_{pk}(\boldsymbol{x}, \boldsymbol{y}'_c) p(\boldsymbol{y}'_c|\boldsymbol{x}) \tag{7}$$

and expose the intuitive form of a difference between the expectation of a sufficient statistic according to the empiric distribution and the expectation according to the model distribution. The latter term requires marginal probabilities for each clique c, denoted by $p(\boldsymbol{y}'_c|\boldsymbol{x})$. Inference on the graphical model G (see sec 5.2) is needed to compute these.

Depending on the structure of G, inference can be very expensive. In order to speed up parameter estimation, which requires inference to be performed for every training example and for every iteration of the gradient-based method, alternatives to MLE have been proposed that do not require inference. We show here a factor-based variant of pseudolikelihood as proposed by Sanner et al. (2007):

$$\ell_p(\boldsymbol{\theta}) = \sum_{C_p \in \mathcal{C}} \sum_{\boldsymbol{\phi}_c \in C_p} \log p(\boldsymbol{y}_c|\boldsymbol{x}, MB(\boldsymbol{\phi}_c)) \tag{8}$$

where the factors are conditioned on the Markov blanket, denoted by MB[6]. The gradient of (8) can be computed similar to (7), except that the marginals $p_c(\boldsymbol{y}'_c|\boldsymbol{x})$ are also conditioned on the Markov blanket, i.e., $p_c(\boldsymbol{y}'_c|\boldsymbol{x}, MB(\boldsymbol{\phi}_c))$. Due to its dependence on the Markov blanket of factors, pseudolikelihood cannot be applied to prediction, but only to parameter estimation, where the "true" assignment of a blanket is known.

5.1.1 Regularization

We employ a Gaussian prior for training of CRFs in order to avoid overfitting. Hence, if $f(\boldsymbol{\theta})$ is the original objective function (e.g., log-likelihood or log-pseudolikelihood), we optimize a penalized version $f'(\boldsymbol{\theta})$ instead, such that:

$$f'(\boldsymbol{\theta}) = f(\boldsymbol{\theta}) - \sum_{k=1}^{|\boldsymbol{\theta}|} \frac{\lambda_k^2}{2\sigma^2} \quad \text{and} \quad \frac{\partial f'}{\partial \lambda_k} = \frac{\partial f}{\partial \lambda_k} - \frac{\lambda_k}{\sigma^2}.$$

The tuning parameter σ^2 determines the strength of the penalty; lower values lead to less overfitting. Gaussian priors are a common choice for parameter estimation of log-linear models (cf. Sutton and McCallum (2007)).

5.2 Inference

Inference on a graphical model G is needed to efficiently compute the normalization term $Z(\boldsymbol{x})$ and marginals $p_c(\boldsymbol{y}'_c|\boldsymbol{x})$ for MLE, cf. equation (6).

Using belief propagation (Yedidia et al., 2003), more precisely its sum-product variant, we can compute the beliefs for all cliques $c \in G$. In a tree-shaped graphical model G, these beliefs correspond exactly to the marginal probabilities $p_c(\boldsymbol{y}'_c|\boldsymbol{x})$. However, if the graph contains cycles, so-called loopy belief propagation must be performed. The message updates are then re-iterated according to some schedule until the messages converge. We use a TRP schedule as described by Wainwright et al. (2002). The resulting beliefs are then only approximations to the true marginals. Moreover, loopy belief propagation is not guaranteed to terminate in general – we investigate this phenomenon in section 6.5.

With regard to the normalization term $Z(\boldsymbol{x})$, as equation (2) shows, naive computation requires summing over all assignments of $\boldsymbol{y}$. This is too expensive to be practical. Fortunately, belief propagation produces an alternative factorization of $p(\boldsymbol{y}|\boldsymbol{x})$; i.e., the conditional distribution defining the CRF can be expressed in terms of the marginals gained during sum-product belief propagation. This representation does not require any additional normalization, so $Z(\boldsymbol{x})$ need not be computed.

[5]Recently, stochastic gradient descent methods such as Online LBFGS (Schraudolph et al., 2007) have been shown to perform competitively.

[6]Here, the Markov blanket of a factor $\boldsymbol{\phi}_c$ denotes the set of variables occurring in factors that share variables with $\boldsymbol{\phi}_c$, non-inclusive of the variables of $\boldsymbol{\phi}_c$

5.3 Prediction

Once the parameters $\boldsymbol{\theta}$ have been estimated from training data, a CRF can be used to predict the labels of unknown examples. The goal is to find:

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}'}{\operatorname{argmax}} \left(p(\boldsymbol{y}'|\boldsymbol{x};\boldsymbol{\theta}) \right) \tag{9}$$

i.e., the assignment of $\boldsymbol{y}$ that maximizes the conditional probability of the CRF. Again, naive computation of (9) is intractable. However, the max-product variant of loopy belief propagation can be applied to approximately find the MAP assignment of $\boldsymbol{y}$ (max-product can be seen as a generalization of the well-known Viterbi algorithm to graphical models).

For structure recognition in medical reports, we employ a post-processing step after label prediction with the CRF model. As in Jancsary (2008), this step enforces the constraints of the `BIO` notation and applies some trivial non-local heuristics that guarantee a consistent global view of the resulting structure.

6 Experiments and Results

For evaluation, we generally performed 3-fold cross-validation for all performance measures. We created training data from the reports in C_{COR} so as to simulate a scenario under ideal conditions, i.e., perfect speech recognition and proper dictation of punctuation and headings, without hesitation or repetitions. In contrast, the data from C_{RCG} reflects real-life conditions, with a wide variety of speech recognition error rates and speakers frequently hesitating, repeating themselves and omitting punctuation and/or headings.

Depending on the experiment, two different subsets of the two corpora were considered:

- $C_{\{COR,RCG\}\text{-}ALL}$: All 2007 reports were used, resulting in 1338 training examples and 669 testing examples at each CV-iteration.
- $C_{\{COR,RCG\}\text{-}BEST}$: The corpus was restricted to those 1002 reports that yielded the lowest word error rate during alignment (see section 4.2). Each CV-iteration hence amounts to 668 training examples and 334 testing examples.

From the crossvalidation runs, a 95%-confidence interval for each measure was estimated as follows:

$$\bar{Y} \pm t_{(\alpha/2,N-1)} \frac{s}{\sqrt{N}} = \bar{Y} \pm t_{(0.025,2)} \frac{s}{\sqrt{3}} \tag{10}$$

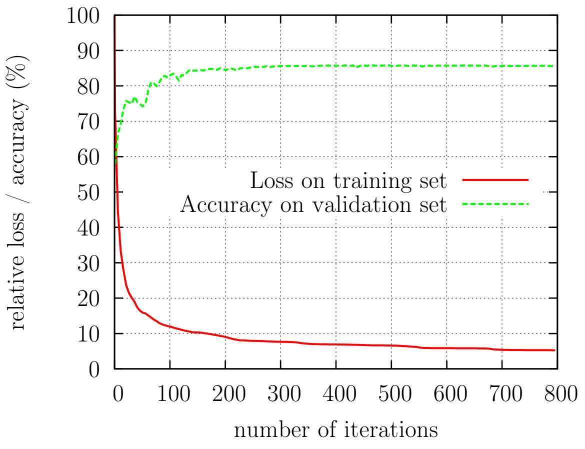


Fig. 5: Accuracy vs. loss function on $C_{RCG\text{-}ALL}$

where $\bar{Y}$ is the sample mean, s is the sample standard deviation, N is the sample size (3), α is the desired significance level (0.05) and $t_{(\alpha/2,N-1)}$ is the upper critical value of the t-distribution with $N-1$ degrees of freedom. The confidence intervals are indicated in the $\pm$ column of tables 1, 2 and 3.

For CRF training, we minimized the penalized, negative log-pseudolikelihood using LBFGS with $m = 3$. The variance of the Gaussian prior was set to $\sigma^2 = 1000$. All supported features were used for univariate factors, while the bivariate factors within chains and between chains were restricted to bias weights. For testing, loopy belief propagation with a TRP schedule was used in order to determine the maximum a posteriori (MAP) assignment. We use *VieCRF*, our own implementation of factorial CRFs, which is freely available at the author's homepage[7].

6.1 Analysis of training progress

In order to determine the number of required training iterations, an experiment was performed that compares the progress of the Accuracy measure on a validation set to the progress of the loss function on a training set. The data was randomly split into a training set (2/3 of the instances) and a validation set. Accuracy on the validation set was computed using the intermediate CRF parameters $\boldsymbol{\theta}_t$ every 5 iterations of LBFGS. The resulting plot (figure 5) demonstrates that the progress of the loss function corresponds well to that of the Accuracy measure,

[7] `http://www.ofai.at/~jeremy.jancsary/`

Estimated Accuracies		
	Acc.	±
Average	97.24%	0.33
Chain 0	99.64%	0.04
Chain 1	95.48%	0.55
Chain 2	96.61%	0.68
Joint	92.51%	0.97

(a) $C_{COR\text{-}ALL}$

Estimated Accuracies		
	Acc.	±
Average	86.36%	0.80
Chain 0	91.74%	0.16
Chain 1	85.90%	1.25
Chain 2	81.45%	2.14
Joint	69.19%	1.93

(b) $C_{RCG\text{-}ALL}$

Table 1: Accuracy on the full corpus

Estimated Accuracies		
	Acc.	±
Average	96.48%	0.82
Chain 0	99.55%	0.08
Chain 1	94.64%	0.23
Chain 2	95.25%	2.16
Joint	90.65%	2.15

(a) $C_{COR\text{-}BEST}$

Estimated Accuracies		
	Acc.	±
Average	87.73%	2.07
Chain 0	93.77%	0.68
Chain 1	87.59%	1.79
Chain 2	81.81%	3.79
Joint	70.91%	4.50

(b) $C_{RCG\text{-}BEST}$

Table 2: Accuracy on a high-quality subset

thus an "early stopping" approach might be tempting to cut down on training times. However, during earlier stages of training, the CRF parameters seem to be strongly biased towards high-frequency labels, so other measures such as macro-averaged F1 might suffer from early stopping. Hence, we decided to allow up to 800 iterations of LBFGS.

6.2 Accuracy of structure prediction

Table 1 shows estimated accuracies for $C_{COR\text{-}ALL}$ and $C_{RCG\text{-}ALL}$. Overall, high accuracy ($> 97\%$) can be achieved on $C_{COR\text{-}ALL}$, showing that the approach works very well under ideal conditions. Performance is still fair on the noisy data ($C_{RCG\text{-}ALL}$; $\text{Accuracy} > 86\%$). It should be noted that the labels are unequally distributed, especially in *chain 0* (there are very few BEGIN labels). Thus, the baseline is substantially high for this chain, and other measures may be better suited for evaluating segmentation quality (cf. section 6.4).

6.3 On the effect of noisy training data

Measuring the effect of the imprecise reference annotation of C_{RCG} is difficult without a corresponding, manually created golden standard. However, to get a feeling for the impact of the noise induced by speech recognition errors and sloppy dictation on the quality of the semi-automatically generated annotation, we conducted an experiment with subsets $C_{COR\text{-}BEST}$ and $C_{RCG\text{-}BEST}$. The results are shown in table 2. Comparing these results to table 1, one can see that overall accuracy *decreased* for $C_{COR\text{-}BEST}$, whereas we see an *increase* for $C_{RCG\text{-}BEST}$. This effect can be attributed to two different phenomena:

- In $C_{COR\text{-}BEST}$, no quality gains in the annotation could be expected. The smaller number of training examples therefore results in lower accuracy.
- Fewer speech recognition errors and more consistent dictation in $C_{RCG\text{-}BEST}$ allow for better alignment and thus a better reference annotation. This increases the actual prediction performance and, furthermore, reduces the number of label predictions that are erroneously counted as a misprediction.

Thus, it is to be expected that manual correction of the automatically created annotation results in significant performance gains. Preliminary annotation experiments have shown that this is indeed the case.

Estimated WD		
	WD	±
Chain 0	0.007	0.000
Chain 1	0.050	0.007
Chain 2	0.015	0.001

(a) $C_{COR\text{-}ALL}$

Estimated WD		
	WD	±
Chain 0	0.193	0.008
Chain 1	0.149	0.005
Chain 2	0.118	0.013

(b) $C_{RCG\text{-}ALL}$

Table 3: Per-chain $\mathrm{WindowDiff}$ on the full corpus

6.4 Segmentation quality

$\mathrm{Accuracy}$ is not the best measure to assess segmentation quality, therefore we also conducted experiments using the $\mathrm{WindowDiff}$ measure as proposed by Pevzner and Hearst (2002). $\mathrm{WindowDiff}$ returns 0 in case of a perfect segmentation; 1 is the worst possible score. However, it only takes into account segment boundaries and disregards segment types. Table 3 shows the $\mathrm{WindowDiff}$ scores for $C_{COR\text{-}ALL}$ and $C_{RCG\text{-}ALL}$. Overall, the scores are quite good and are consistently below 0.2. Furthermore, $C_{RCG\text{-}ALL}$ scores do not suffer as badly from inaccurate reference annotation, since "near misses" are penalized less strongly.

	Converged (%)	Iterations (∅)
$C_{COR\text{-}ALL}$	0.999	15.4
$C_{RCG\text{-}ALL}$	0.911	66.5
$C_{COR\text{-}BEST}$	0.999	14.2
$C_{RCG\text{-}BEST}$	0.971	37.5

Table 4: Convergence behaviour of loopy BP

6.5 Convergence of loopy belief propagation

In section 5.2, we mentioned that loopy BP is not guaranteed to converge in a finite number of iterations. Since we optimize pseudolikelihood for parameter estimation, we are not affected by this limitation in the training phase. However, we use loopy BP with a TRP schedule during testing, so we must expect to encounter non-convergence for some examples. Theoretical results on this topic are discussed by Heskes (2004). We give here an empirical observation of convergence behaviour of loopy BP in our setting; the maximum number of iterations of the TRP schedule was restricted to 1,000. Table 4 shows the percentage of examples converging within this limit and the average number of iterations required by the converging examples, broken down by the different corpora. From these results, we conclude that there is a connection between the quality of the annotation and the convergence behaviour of loopy BP. In practice, even though loopy BP didn't converge for some examples, the solutions after 1,000 iterations where satisfactory.

7 Conclusion and Outlook

We have presented a framework which allows for identification of structure in report dictations, such as sentence boundaries, paragraphs, enumerations, (sub)sections, and various other structural elements; even if no explicit clues are dictated. Furthermore, meaningful types are automatically assigned to subsections and sections, allowing – for instance – to automatically assign headings, if none were dictated.

For the preparation of training data a mechanism has been presented that exploits the potential of parallel corpora for automatic annotation of data. Using manually edited formatted reports and the corresponding raw output of ASR, reference annotation can be generated that is suitable for learning to identify structure in ASR output.

For the structure recognition task, a CRF framework has been employed and multiple experiments have been performed, confirming the practicability of the approach presented here.

One result deserving further investigation is the effect of noisy annotation. We have shown that segmentation results improve when fewer errors are present in the automatically generated annotation. Thus, manual correction of the reference annotation will yield further improvements.

Finally, the framework presented in this paper opens up exciting possibilities for future work. In particular, we aim at automatically transforming report dictations into properly formatted and rephrased reports that conform to the requirements of the relevant domain. Such tasks are greatly facilitated by the explicit knowledge gained during structure recognition.

Acknowledgments

The work presented here has been carried out in the context of the Austrian KNet competence network COAST. We gratefully acknowledge funding by the Austrian Federal Ministry of Economics and Labour, and ZIT Zentrum fuer Innovation und Technologie, Vienna. The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry for Transport, Innovation, and Technology and by the Austrian Federal Ministry for Science and Research.

Furthermore, we would like to thank our anonymous reviewers for many insightful comments that helped us improve this paper.

References

ASTM International. 2002. ASTM E2184-02: Standard specification for healthcare document formats.

Freddy Choi. 2000. Advances in domain independent linear text segmentation. In *Proceedings of the first conference on North American chapter of the Association for Computation Linguistics*, pages 26–33.

C. Fellbaum. 1998. *WordNet: an electronic lexical database*. MIT Press, Cambridge, MA.

Marti A. Hearst. 1997. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1):36–47.

Tom Heskes. 2004. On the uniqueness of loopy belief propagation fixed points. *Neural Comput.*, 16(11):2379–2413.

Martin Huber, Jeremy Jancsary, Alexandra Klein, Johannes Matiasek, and Harald Trost. 2006. Mismatch interpretation by semantics-driven alignment. In *Proceedings of KONVENS '06*.

Jeremy M. Jancsary. 2008. Recognizing structure in report transcripts. Master's thesis, Vienna University of Technology.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*.

S. Lamprier, T. Amghar, B. Levrat, and F. Saubion. 2008. Toward a more global and coherent segmentation of texts. *Applied Artificial Intelligence*, 23:208–234, March.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710.

D. A. B. Lindberg, B. L. Humphreys, and A. T. McCray. 1993. The Unified Medical Language System. *Methods of Information in Medicine*, 32:281–291.

Evgeny Matsuov. 2003. Statistical methods for text segmentation and topic detection. Master's thesis, Rheinisch-Westfälische Technische Hochschule Aachen.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Flexible text segmentation with structured multilabel classification. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 987–994.

Jorge Nocedal. 1980. Updating Quasi-Newton matrices with limited storage. *Mathematics of Computation*, 35:773–782.

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

Lawrence Philips. 1990. Hanging on the metaphone. *Computer Language*, 7(12).

L. R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–286, February.

Scott Sanner, Thore Graepel, Ralf Herbrich, and Tom Minka. 2007. Learning CRFs with hierarchical features: An application to go. International Conference on Machine Learning (ICML) workshop.

Nicol N. Schraudolph, Jin Yu, and Simon Günter. 2007. A stochastic Quasi-Newton Method for online convex optimization. In *Proceedings of 11th International Conference on Artificial Intelligence and Statistics*.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

Charles Sutton and Andrew McCallum. 2005. Composition of Conditional Random Fields for transfer learning. In *Proceedings of Human Language Technologies / Empirical Methods in Natural Language Processing (HLT/EMNLP)*.

Charles Sutton and Andrew McCallum. 2007. An introduction to Conditional Random Fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press.

Martin Wainwright, Tommi Jaakkola, and Alan S. Willsky. 2002. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Transactions on Information Theory*, 49(5).

Ming Ye and Paul Viola. 2004. Learning to parse hierarchical lists and outlines using Conditional Random Fields. In *Proceedings of the Ninth International Workshop on Frontiers in Handwriting Recognition (IWFHR'04)*, pages 154–159. IEEE Computer Society.

Jonathan S. Yedidia, William T. Freeman, and Yair Weiss, 2003. *Understanding Belief Propagation and its Generalizations, Exploring Artificial Intelligence in the New Millennium*, chapter 8, pages 236–239. Science & Technology Books, January.

It's a Contradiction—No, it's Not: A Case Study using Functional Relations

Alan Ritter, Doug Downey, Stephen Soderland and Oren Etzioni
Turing Center
Department of Computer Science and Engineering
University of Washington
Box 352350
Seattle, WA 98195, USA
{aritter,ddowney,soderlan,etzioni}@cs.washington.edu

Abstract

Contradiction Detection (CD) in text is a difficult NLP task. We investigate CD over functions (*e.g.*, BornIn(Person)=Place), and present a domain-independent algorithm that automatically discovers phrases denoting functions with high precision. Previous work on CD has investigated hand-chosen sentence pairs. In contrast, we automatically harvested from the Web pairs of sentences that *appear* contradictory, but were surprised to find that most pairs are in fact consistent. For example, "Mozart was born in Salzburg" does *not* contradict "Mozart was born in Austria" despite the functional nature of the phrase "was born in". We show that background knowledge about meronyms (*e.g.*, Salzburg is in Austria), synonyms, functions, and more is essential for success in the CD task.

1 Introduction and Motivation

Detecting contradictory statements is an important and challenging NLP task with a wide range of potential applications including analysis of political discourse, of scientific literature, and more (de Marneffe et al., 2008; Condoravdi et al., 2003; Harabagiu et al., 2006). De Marneffe *et al.* present a model of CD that defines the task, analyzes different types of contradictions, and reports on a CD system. They report 23% precision and 19% recall at detecting contradictions in the RTE-3 data set (Voorhees, 2008). Although RTE-3 contains a wide variety of contradictions, it does not reflect the prevalence of *seeming* contradictions and the paucity of *genuine* contradictions, which we have found in our corpus.

1.1 Contradictions and World Knowledge

Our paper is motivated in part by de Marneffe *et al.*'s work, but with some important differences. First, we introduce a simple logical foundation for the CD task, which suggests that extensive world knowledge is essential for building a domain-independent CD system. Second, we automatically generate a large corpus of *apparent* contradictions found in arbitrary Web text. We show that most of these apparent contradictions are actually consistent statements due to meronyms (Alan Turing was born in London *and* in England), synonyms (George Bush is married to *both* Mrs. Bush and Laura Bush), hypernyms (Mozart died of *both* renal failure and kidney disease), and reference ambiguity (one John Smith was born in 1997 and a *different* John Smith in 1883). Next, we show how background knowledge enables a CD system to discard seeming contradictions and focus on genuine ones.

De Marneffe *et al.* introduced a typology of contradiction in text, but focused primarily on contradictions that can be detected from linguistic evidence (*e.g.* negation, antonymy, and structural or lexical disagreements). We extend their analysis to a class of contradictions that can only be detected utilizing background knowledge. Consider for example the following sentences:

1) "Mozart was born in Salzburg."
2) "Mozart was born in Vienna."
3) "Mozart visited Salzburg."
4) "Mozart visited Vienna."

Sentences 1 & 2 are contradictory, but 3 & 4 are not. Why is that? The distinction is not syntactic. Rather, sentences 1 and 2 are contradictory because the relation expressed by the phrase "was born in" can be characterized here as a *function* from people's names to their unique birthplaces. In contrast, "visited" does not denote a functional relation.[1]

We cannot assume that a CD system knows, in advance, all the functional relations that might appear in a corpus. Thus, a central challenge for a function-based CD system is to determine which relations are functional based on a corpus. Intuitively, we might expect that "functional phrases" such as "was born in" would typically map person names to unique place names, making function detection easy. But, in fact, function detection is surprisingly difficult because name ambiguity (*e.g.*, John Smith), common nouns (*e.g.*, "dad" or "mom"), definite descriptions (*e.g.*, "the president"), and other linguistic phenomena can mask functions in text. For example, the two sentences "John Smith was born in 1997." and "John Smith was born in 1883." can be viewed as either evidence that "was born in" does not denote a function or, alternatively, that "John Smith" is ambiguous.

1.2 A CD System Based on Functions

We report on the AUCONTRAIRE CD system, which addresses each of the above challenges. First, AUCONTRAIRE identifies "functional phrases" statistically (Section 3). Second, AUCONTRAIRE uses these phrases to automatically create a large corpus of apparent contradictions (Section 4.2). Finally, AUCONTRAIRE sifts through this corpus to find genuine contradictions using knowledge about synonymy, meronymy, argument types, and ambiguity (Section 4.3).

Instead of analyzing sentences directly, AUCONTRAIRE relies on the TEXTRUNNER Open Information Extraction system (Banko et al., 2007; Banko and Etzioni, 2008) to map each sentence to one or more tuples that represent the entities in the sentences and the relationships between them (*e.g.*, *was_born_in(Mozart,Salzburg)*). Using extracted tuples greatly simplifies the CD task, because numerous syntactic problems (*e.g.*, anaphora, relative clauses) and semantic challenges (*e.g.*, quantification, counterfactuals, temporal qualification) are delegated to TEXTRUNNER or simply ignored. Nevertheless, extracted tuples are a convenient approximation of sentence content, which enables us to focus on function detection and function-based CD.

Our contributions are the following:

- We present a novel model of the Contradiction Detection (CD) task, which offers a simple logical foundation for the task and emphasizes the central role of background knowledge.
- We introduce and evaluate a new EM-style algorithm for detecting whether phrases denote functional relations and whether nouns (*e.g.*, "dad") are ambiguous, which enables a CD system to identify functions in arbitrary domains.
- We automatically generate a corpus of seeming contradictions from Web text, and report on a set of experiments over this corpus, which provide a baseline for future work on statistical function identification and CD. [2]

2 A Logical Foundation for CD

On what basis can a CD system conclude that two statements T and H are contradictory? Logically, contradiction holds when $T \models \neg H$. As de Marneffe *et al.* point out, this occurs when T and H contain antonyms, negation, or other lexical elements that suggest that T and H are directly contradictory. But other types of contradictions can only be detected with the help of a body of background knowledge K: In these cases, T and H *alone* are mutually consistent. That is,

$$T \not\models \neg H \wedge H \not\models \neg T$$

A contradiction between T and H arises only in the context of K. That is:

$$((K \wedge T) \models \neg H) \vee ((K \wedge H) \models \neg T)$$

Consider the example of Mozart's birthplace in the introduction. To detect a contradiction, a CD system must know that A) "Mozart" refers to the same entity in both sentences, that B) "was born in" denotes a functional relation, and that C) Vienna and Salzburg are inconsistent locations.

[1] Although we focus on function-based CD in our case study, we believe that our observations apply to other types of CD as well.

[2] The corpus is available at `http://www.cs.washington.edu/research/aucontraire/`

Of course, world knowledge, and reasoning about text, are often uncertain, which leads us to associate probabilities with a CD system's conclusions. Nevertheless, the knowledge base K is essential for CD.

We now turn to a probabilistic model that helps us simultaneously estimate the functionality of relations (B in the above example) and ambiguity of argument values (A above). Section 4 describes the remaining components of AUCONTRAIRE.

3 Detecting Functionality and Ambiguity

This section introduces a formal model for computing the probability that a phrase denotes a function based on a set of extracted tuples. An extracted tuple takes the form $R(x, y)$ where (roughly) x is the subject of a sentence, y is the object, and R is a phrase denoting the relationship between them. If the relation denoted by R is functional, then typically the object y is a function of the subject x. Thus, our discussion focuses on this possibility, though the analysis is easily extended to the symmetric case.

Logically, a relation R is functional in a variable x if it maps it to a unique variable y: $\forall x, y_1, y_2 \; R(x, y_1) \wedge R(x, y_2) \Rightarrow y_1 = y_2$. Thus, given a large random sample of ground instances of R, we could detect with high confidence whether R is functional. In text, the situation is far more complex due to ambiguity, polysemy, synonymy, and other linguistic phenomena. Deciding whether R is functional becomes a probabilistic assessment based on aggregated textual evidence.

The main evidence that a relation $R(x, y)$ is functional comes from the distribution of y values for a given x value. If R denotes a function and x is unambiguous, then we expect the extractions to be predominantly a single y value, with a few outliers due to noise. We aggregate the evidence that R is *locally* functional for a particular x value to assess whether R is *globally* functional for all x.

We refer to a set of extractions with the same relation R and argument x as a *contradiction set* $R(x, \cdot)$. Figure 1 shows three example contradiction sets. Each example illustrates a situation commonly found in our data. Example A in Figure 1 shows strong evidence for a functional relation. 66 out of 70 TEXTRUNNER extractions for *was_born_in (Mozart, PLACE)* have the same y value. An ambiguous x argument, however, can make a functional relation *appear* non-functional. Example B depicts a distribution of y values that appears less functional due to the fact that "John Adams" refers to multiple, distinct real-world individuals with that name. Finally, example C exhibits evidence for a non-functional relation.

A. was_born_in(Mozart, PLACE):
   Salzburg(66), Germany(3), Vienna(1)

B. was_born_in(John Adams, PLACE):
   Braintree(12), Quincy(10), Worcester(8)

C. lived_in(Mozart, PLACE):
   Vienna(20), Prague(13), Salzburg(5)

Figure 1: Functional relations such as example A have a different distribution of y values than non-functional relations such as C. However, an ambiguous x argument as in B, can make a functional relation *appear* non-functional.

3.1 Formal Model of Functions in Text

To decide whether R is functional in x for all x, we first consider how to detect whether R is *locally functional* for a particular value of x. The local functionality of R with respect to x is the probability that R is functional estimated solely on evidence from the distribution of y values in a contradiction set $R(x, \cdot)$.

To decide the probability that R is a function, we define global functionality as the average local functionality score for each x, weighted by the probability that x is unambiguous. Below, we outline an EM-style algorithm that alternately estimates the probability that R is functional and the probability that x is ambiguous.

Let R^*_x indicate the event that the relation R is locally functional for the argument x, and that x is locally unambiguous for R. Also, let D indicate the set of observed tuples, and define $D_{R(x,\cdot)}$ as the multi-set containing the frequencies for extractions of the form $R(x, \cdot)$. For example the distribution of extractions from Figure 1 for example A is

$D_{\mathrm{W}as\_born\_in(\mathrm{M}ozart,\cdot)} = \{66, 3, 1\}$.

Let θ^f_R be the probability that $R(x, \cdot)$ is locally functional for a random x, and let Θ^f be the vector of these parameters across all relations R. Likewise, θ^u_x represents the probability that x is locally unambiguous for random R, and Θ^u the vector for all x.

We wish to determine the *maximum a posteriori* (MAP) functionality and ambiguity parameters given the observed data D, that is $\arg\max_{\Theta^f,\Theta^u} P(\Theta^f, \Theta^u|D)$. By Bayes Rule:

$$P(\Theta^f, \Theta^u|D) = \frac{P(D|\Theta^f, \Theta^u)P(\Theta^f, \Theta^u)}{P(D)} \quad (1)$$

We outline a generative model for the data, $P(D|\Theta^f, \Theta^u)$. Let us assume that the event R^*_x depends only on θ^f_R and θ^u_x, and further assume that given these two parameters, local ambiguity and local functionality are conditionally independent. We obtain the following expression for the probability of R^*_x given the parameters:

$$P(R^*_x|\Theta^f, \Theta^u) = \theta^f_R\theta^u_x$$

We assume each set of data $D_{R(x,\cdot)}$ is generated independently of all other data and parameters, given R^*_x. From this and the above we have:

$$P(D|\Theta^f, \Theta^u) = \prod_{R,x} \Big(P(D_{R(x,\cdot)}|R^*_x)\theta^f_R\theta^u_x + P(D_{R(x,\cdot)}|\neg R^*_x)(1-\theta^f_R\theta^u_x)\Big) \quad (2)$$

These independence assumptions allow us to express $P(D|\Theta^f, \Theta^u)$ in terms of distributions over $D_{R(x,\cdot)}$ given whether or not R^*_x holds. We use the URNS model as described in (Downey et al., 2005) to estimate these probabilities based on binomial distributions. In the single-urn URNS model that we utilize, the extraction process is modeled as draws of labeled balls from an urn, where the labels are either correct extractions or errors, and different labels can be repeated on varying numbers of balls in the urn.

Let $k = \max D_{R(x,\cdot)}$, and let $n = \sum D_{R(x,\cdot)}$; we will approximate the distribution over $D_{R(x,\cdot)}$ in terms of k and n. If $R(x,\cdot)$ is locally functional and unambiguous, there is exactly one correct extraction label in the urn (potentially repeated multiple times). Because the probability of correctness tends to increase with extraction frequency, we make the simplifying assumption that the most frequently extracted element is correct.[3] In this case, k is the number of correct extractions, which by the URNS model has a binomial distribution with parameters n and p, where p is the precision of the extraction process. If $R(x,\cdot)$ is *not* locally functional and unambiguous, then we expect k to typically take on smaller values. Empirically, the underlying frequency of the most frequent element in the $\neg R^*_x$ case tends to follow a Beta distribution.

Under the model, the probability of the evidence given R^*_x is:

$$P(D_{R(x,\cdot)}|R^*_x) \approx P(k,n|R^*_x) = \binom{n}{k} p^k(1-p)^{n-k}$$

And the probability of the evidence given $\neg R^*_x$ is:

$$P(D_{R(x,\cdot)}|\neg R^*_x) \approx P(k,n|\neg R^*_x)$$
$$= \tbinom{n}{k} \int_0^1 \frac{p'^{k+\alpha_f-1}(1-p')^{n+\beta_f-1-k}}{B(\alpha_f,\beta_f)} dp'$$
$$= \frac{\binom{n}{k}\Gamma(n-k+\beta_f)\Gamma(\alpha_f+k)}{B(\alpha_f,\beta_f)\Gamma(\alpha_f+\beta_f+n)} \quad (3)$$

where n is the sum over $D_{R(x,\cdot)}$, Γ is the Gamma function and B is the Beta function. α_f and β_f are the parameters of the Beta distribution for the $\neg R^*_x$ case. These parameters and the prior distributions are estimated empirically, based on a sample of the data set of relations described in Section 5.1.

3.2 Estimating Functionality and Ambiguity

Substituting Equation 3 into Equation 2 and applying an appropriate prior gives the probability of parameters Θ^f and Θ^u given the observed data D. However, Equation 2 contains a large product of sums—with two independent vectors of coefficients, Θ^f and Θ^u—making it difficult to optimize analytically.

If we knew which arguments were ambiguous, we would ignore them in computing the functionality of a relation. Likewise, if we knew which relations were non-functional, we would ignore them in computing the ambiguity of an argument. Instead, we initialize the Θ^f and Θ^u arrays randomly, and then execute an algorithm similar to Expectation-Maximization (EM) (Dempster et al., 1977) to arrive at a high-probability setting of the parameters.

Note that if Θ^u is fixed, we can compute the expected fraction of locally unambiguous arguments x for which R is locally functional, using $D_{R(x',\cdot)}$ and

[3]As this assumption is invalid when there is not a unique maximal element, we default to the prior $P(R^*_x)$ in that case.

Equation 3. Likewise, for fixed Θ^f, for any given x we can compute the expected fraction of locally functional relations R that are locally unambiguous for x.

Specifically, we repeat until convergence:

1. Set $\theta^f_R = \frac{1}{s_R}\sum_x P(R^*_x|D_{R(x,\cdot)})\theta^u_x$ for all R.
2. Set $\theta^u_x = \frac{1}{s_x}\sum_R P(R^*_x|D_{R(x,\cdot)})\theta^f_R$ for all x.

In both steps above, the sums are taken over only those x or R for which $D_{R(x,\cdot)}$ is non-empty. Also, the normalizer $s_R = \sum_x \theta^u_x$ and likewise $s_x = \sum_R \theta^f_R$.

As in standard EM, we iteratively update our parameter values based on an expectation computed over the unknown variables. However, we alternately optimize two disjoint sets of parameters (the functionality and ambiguity parameters), rather than just a single set of parameters as in standard EM. Investigating the optimality guarantees and convergence properties of our algorithm is an item of future work.

By iteratively setting the parameters to the expectations in steps 1 and 2, we arrive at a good setting of the parameters. Section 5.2 reports on the performance of this algorithm in practice.

4 System Overview

AuContraire identifies phrases denoting functional relations and utilizes these to find contradictory assertions in a massive, open-domain corpus of text.

AuContraire begins by finding extractions of the form $R(x, y)$, and identifies a set of relations R that have a high probability of being functional. Next, AuContraire identifies contradiction sets of the form $R(x, \cdot)$. In practice, most contradiction sets turned out to consist overwhelmingly of seeming contradictions—assertions that do not actually contradict each other for a variety of reasons that we enumerate in section 4.3. Thus, a major challenge for AuContraire is to tease apart which pairs of assertions in $R(x, \cdot)$ represent genuine contradictions.

Here are the main components of AuContraire as illustrated in Figure 2:

Extractor: Create a set of extracted assertions $\mathcal{E}$ from a large corpus of Web pages or other documents. Each extraction $R(x, y)$ has a probability p of being correct.

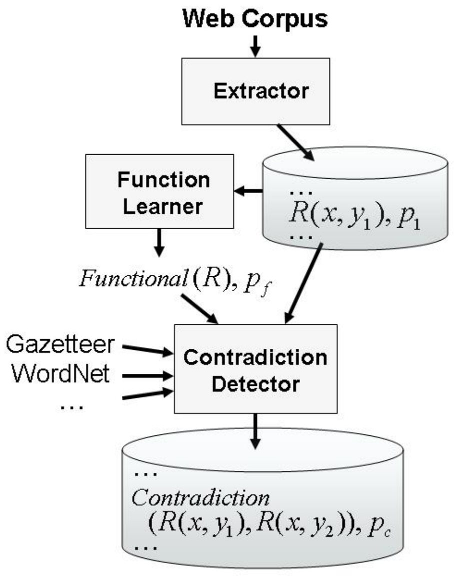


Figure 2: AuContraire architecture

Function Learner: Discover a set of functional relations $\mathcal{F}$ from among the relations in $\mathcal{E}$. Assign to each relation in $\mathcal{F}$ a probability p_f that it is functional.

Contradiction Detector: Query $\mathcal{E}$ for assertions with a relation R in $\mathcal{F}$, and identify sets $\mathcal{C}$ of potentially contradictory assertions. Filter out seeming contradictions in $\mathcal{C}$ by reasoning about synonymy, meronymy, argument types, and argument ambiguity. Assign to each potential contradiction a probability p_c that it is a genuine contradiction.

4.1 Extracting Factual Assertions

AuContraire needs to explore a large set of factual assertions, since genuine contradictions are quite rare (see Section 5). We used a set of extractions $\mathcal{E}$ from the Open Information Extraction system, TextRunner (Banko et al., 2007), which was run on a set of 117 million Web pages.

TextRunner does not require a pre-defined set of relations, but instead uses shallow linguistic analysis and a domain-independent model to identify phrases from the text that serve as relations and phrases that serve as arguments to that relation. TextRunner creates a set of extractions in a single pass over the Web page collection and provides an index to query the vast set of extractions.

Although its extractions are noisy, TextRunner provides a probability that the extractions are correct, based in part on corroboration of facts from different Web pages (Downey et al., 2005).

4.2 Finding Potential Contradictions

The next step of AuContraire is to find contradiction sets in $\mathcal{E}$.

We used the methods described in Section 3 to estimate the functionality of the most frequent relations in $\mathcal{E}$. For each relation R that AuContraire has judged to be functional, we identify contradiction sets $R(x, \cdot)$, where a relation R and domain argument x have multiple range arguments y.

4.3 Handling Seeming Contradictions

For a variety of reasons, a pair of extractions $R(x, y_1)$ and $R(x, y_2)$ may not be actually contradictory. The following is a list of the major sources of false positives—pairs of extractions that are not genuine contradictions, and how they are handled by AuContraire. The features indicative of each condition are combined using Logistic Regression, in order to estimate the probability that a given pair, $\{R(x, y_1), R(x, y_2)\}$ is a genuine contradiction.

Synonyms: The set of potential contradictions *died_from(Mozart,·)* may contain assertions that Mozart died from *renal failure* and that he died from *kidney failure*. These are distinct values of y, but do not contradict each other, as the two terms are synonyms. AuContraire uses a variety of knowledge sources to handle synonyms. WordNet is a reliable source of synonyms, particularly for common nouns, but has limited recall. AuContraire also utilizes synonyms generated by Resolver (Yates and Etzioni, 2007)— a system that identifies synonyms from TextRunner extractions. Additionally, AuContraire uses edit-distance and token-based string similarity (Cohen et al., 2003) between apparently contradictory values of y to identify synonyms.

Meronyms: For some relations, there is no contradiction when y_1 and y_2 share a meronym, *i.e.* "part of" relation. For example, in the set *born_in(Mozart,·)* there is no contradiction between the y values "Salzburg" and "Austria", but "Salzburg" conflicts with "Vienna". Although this is only true in cases where y occurs in an upward monotone context (MacCartney and Manning, 2007), in practice genuine contradictions between y-values sharing a meronym relationship are extremely rare. We therefore simply assigned contradictions between meronyms a probability close to zero. We used the Tipster Gazetteer[4] and WordNet to identify meronyms, both of which have high precision but low coverage.

Argument Typing: Two y values are not contradictory if they are of different argument types. For example, the relation *born_in* can take a date or a location for the y value. While a person can be born in only one year and in only one city, a person can be born in both a year and a city. To avoid such false positives, AuContraire uses a simple named-entity tagger[5] in combination with large dictionaries of person and location names to assign high-level types (person, location, date, other) to each argument. AuContraire filters out extractions from a contradiction set that do not have matching argument types.

Ambiguity: As pointed out in Section 3, false contradictions arise when a single x value refers to multiple real-world entities. For example, if the contradiction set *born_in(John Sutherland, ·)* includes birth years of both 1827 and 1878, is one of these a mistake, or do we have a grandfather and grandson with the same name? AuContraire computes the probability that an x value is unambiguous as part of its Function Learner (see Section 3). An x value can be identified as ambiguous if its distribution of y values is non-functional for multiple functional relations.

If a pair of extractions, $\{R(x, y_1), R(x, y_2)\}$, does not fall into any of the above categories and R is functional, then it is likely that the sentences underlying the extractions are indeed contradictory. We combined the various knowledge sources described above using Logistic Regression, and used 10-fold cross-validation to automatically tune the weights associated with each knowledge source. In addition, the learning algorithm also utilizes the following features:

- Global functionality of the relation, θ_R^f
- Global unambiguity of x, θ_x^u

[4] `http://crl.nmsu.edu/cgi-bin/Tools/CLR/clrcat`

[5] `http://search.cpan.org/~simon/Lingua-EN-NamedEntity-1.1/NamedEntity.pm`

- Local functionality of $R(x, \cdot)$
- String similarity (a combination of token-based similarity and edit-distance) between y_1 and y_2
- The argument types (person, location, date, or other)

The learned model is then used to estimate how likely a potential contradiction $\{R(x, y_1), R(x, y_2)\}$ is to be genuine.

5 Experimental Results

We evaluated several aspects of AUCONTRAIRE: its ability to detect functional relations and to detect ambiguous arguments (Section 5.2); its precision and recall in contradiction detection (Section 5.3); and the contribution of AUCONTRAIRE's key knowledge sources (Section 5.4).

5.1 Data Set

To evaluate AUCONTRAIRE we used TEXTRUNNER's extractions from a corpus of 117 million Web pages. We restricted our data set to the 1,000 most frequent relations, in part to keep the experiments tractable and also to ensure sufficient statistical support for identifying functional relations.

We labeled each relation as functional or not, and computed an estimate of the probability it is functional as described in section 3.2. Section 5.2 presents the results of the Function Learner on this set of relations. We took the top 2% (20 relations) as $\mathcal{F}$, the set of functional relations in our experiments. Out of these, 75% are indeed functional. Some examples include: *was born in*, *died in*, and *was founded by*.

There were 1.2 million extractions for all thousand relations, and about 20,000 extractions in 6,000 contradiction sets for all relations in $\mathcal{F}$.

We hand-tagged 10% of the contradiction sets $R(x, \cdot)$ where $R \in \mathcal{F}$, discarding any sets with over 20 distinct y values since the x argument for that set is almost certainly ambiguous. This resulted in a data set of 567 contradiction sets containing a total of 2,564 extractions and 8,844 potentially contradictory pairs of extractions.

We labeled each of these 8,844 pairs as contradictory or not. In each case, we inspected the original sentences, and if the distinction was unclear, consulted the original source Web pages, Wikipedia articles, and Web search engine results.

In our data set, genuine contradictions over functional relations are surprisingly rare. We found only 110 genuine contradictions in the hand-tagged sample, only 1.2% of the potential contradiction pairs.

5.2 Detecting Functionality and Ambiguity

We ran AUCONTRAIRE's EM algorithm on the thousand most frequent relations. Performance converged after 5 iterations resulting in estimates of the probability that each relation is functional and each x argument is unambiguous. We used these probabilities to generate the precision-recall curves shown in Figure 3.

The graph on the left shows results for functionality, while the graph on the right shows precision at finding unambiguous arguments. The solid lines are results after 5 iterations of EM, and the dashed lines are from computing functionality or ambiguity without EM (*i.e.* assuming uniform values of Θ^c when computing Θ^f and vice versa). The EM algorithm improved results for both functionality and ambiguity, increasing area under curve (AUC) by 19% for functionality and by 31% for ambiguity.

Of course, the ultimate test of how well AUCONTRAIRE can identify functional relations is how well the Contradiction Detector performs on automatically identified functional relations.

5.3 Detecting Contradictions

We conducted experiments to evaluate how well AUCONTRAIRE distinguishes genuine contradictions from false positives.

The bold line in Figure 4 depicts AUCONTRAIRE performance on the distribution of contradictions and seeming contradictions found in actual Web data. The dashed line shows the performance of AUCONTRAIRE on an artificially "balanced" data set that we constructed to contain 50% genuine contradictions and 50% seeming ones.

Previous research in CD presented results on manually selected data sets with a relatively balanced mix of positive and negative instances. As Figure 4 suggests, this is a much easier problem than CD "in the wild". The data gathered from the Web is badly skewed, containing only 1.2% genuine contradictions.

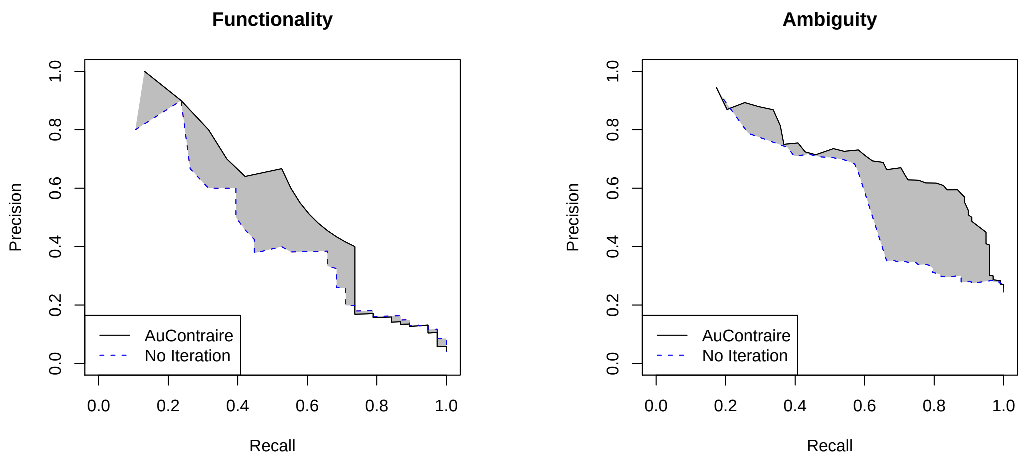


Figure 3: After 5 iterations of EM, AUCONTRAIRE achieves a 19% boost to area under the precision-recall curve (AUC) for functionality detection, and a 31% boost to AUC for ambiguity detection.

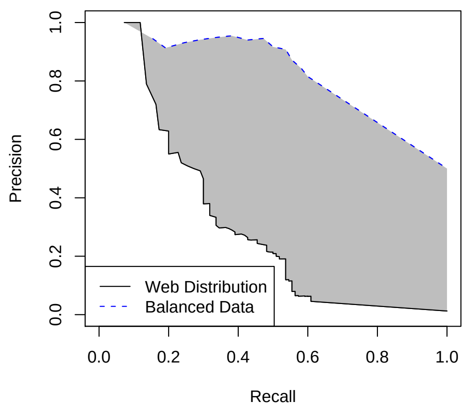


Figure 4: Performance of AUCONTRAIRE at distinguishing genuine contradictions from false positives. The bold line is results on the actual distribution of data from the Web. The dashed line is from a data set constructed to have 50% positive and 50% negative instances.

5.4 Contribution of Knowledge Sources

We carried out an ablation study to quantify how much each knowledge source contributes to AUCONTRAIRE's performance. Since most of the knowledge sources do not apply to numeric argument values, we excluded the extractions where y is a number in this study. As shown in Figure 5, performance of AUCONTRAIRE degrades with no knowledge of synonyms (NS), with no knowledge of meronyms (NM), and especially without argument typing (NT). Conversely, improvements to any of these three components would likely improve the performance of AUCONTRAIRE.

The relatively small drop in performance from no meronyms does not indicate that meronyms are not essential to our task, only that our knowledge sources for meronyms were not as useful as we hoped. The Tipster Gazetteer has surprisingly low coverage for our data set. It contains only 41% of the y values that are locations. Many of these are matches on a different location with the same name, which results in incorrect meronym information. We estimate that a gazetteer with complete coverage would *increase* area under the curve by approximately 40% compared to a system with meronyms from the Tipster Gazetteer and WordNet.

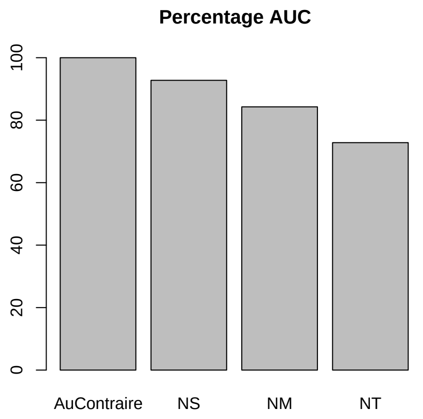


Figure 5: Area under the precision-recall curve for the full AUCONTRAIRE and for AUCONTRAIRE with knowledge removed. NS has no synonym knowledge; NM has no meronym knowledge; NT has no argument typing.

To analyze the errors made by AUCONTRAIRE, we hand-labeled all false-positives at the point of maximum F-score: 29% Recall and 48% Precision.

Figure 6 reveals the central importance of world knowledge for the CD task. About half of the errors (49%) are due to ambiguous x-arguments, which we found to be one of the most persistent obstacles to discovering genuine contradictions. A sizable portion is due to missing meronyms (34%) and missing synonyms (14%), suggesting that lexical resources with broader coverage than WordNet and the Tipster Gazetteer would substantially improve performance. Surprisingly, only 3% are due to errors in the extraction process.

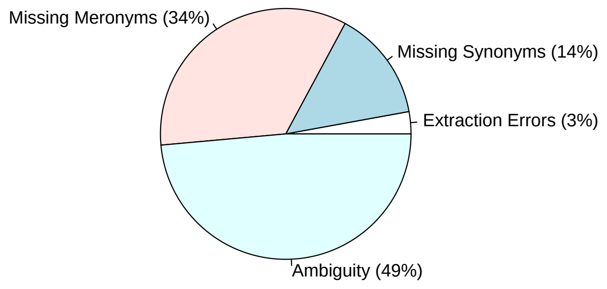


Figure 6: Sources of errors in contradiction detection.

All of our experimental results are based on the automatically discovered set of functions $\mathcal{F}$. We would expect AuContraire's performance to improve substantially if it were given a large set of functional relations as input.

6 Related Work

Condoravdi *et al.* (2003) first proposed contradiction detection as an important NLP task, and Harabagiu *et al.* (2006) were the first to report results on contradiction detection using negation, although their evaluation corpus was a balanced data set built by manually negating entailments in a data set from the Recognizing Textual Entailment conferences (RTE) (Dagan et al., 2005). De Marneffe *et al.* (2008) reported experimental results on a contradiction corpus created by annotating the RTE data sets.

RTE-3 included an optional task, requiring systems to make a 3-way distinction: {entails, contradicts, neither} (Voorhees, 2008). The average performance for contradictions on the RTE-3 was precision 0.11 at recall 0.12, and the best system had precision 0.23 at recall 0.19. We did not run AuContraire on the RTE data sets because they contained relatively few of the "functional contradictions" that AuContraire tackles. On our Web-based data sets, we achieved a precision of 0.62 at recall 0.19, and precision 0.92 at recall 0.51 on the balanced data set. Of course, comparisons across very different data sets are not meaningful, but merely serve to underscore the difficulty of the CD task.

In contrast to previous work, AuContraire is the first to do CD on data automatically extracted from the Web. This is a much harder problem than using an artificially balanced data set, as shown in Figure 4.

Automatic discovery of functional relations has been addressed in the database literature as *Functional Dependency Mining* (Huhtala et al., 1999; Yao and Hamilton, 2008). This focuses on discovering functional relationships between sets of attributes, and does not address the ambiguity inherent in natural language.

7 Conclusions and Future Work

We have described a case study of contradiction detection (CD) based on functional relations. In this context, we introduced and evaluated the AuContraire system and its novel EM-style algorithm for determining whether an arbitrary phrase is functional. We also created a unique "natural" data set of *seeming contradictions* based on sentences drawn from a Web corpus, which we make available to the research community.

We have drawn two key lessons from our case study. First, many seeming contradictions (approximately 99% in our experiments) are not genuine contradictions. Thus, the CD task may be much harder on natural data than on RTE data as suggested by Figure 4. Second, extensive background knowledge is necessary to tease apart seeming contradictions from genuine ones. We believe that these lessons are broadly applicable, but verification of this claim is a topic for future work.

Acknowledgements

This research was supported in part by NSF grants IIS-0535284 and IIS-0312988, ONR grant N00014-08-1-0431 as well as gifts from the Utilika Foundation and Google, and was carried out at the University of Washington's Turing Center.

References

M. Banko and O. Etzioni. 2008. The tradeoffs between traditional and open relation extraction. In *Proceedings of ACL.*

M. Banko, M. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open information extraction from the Web. In *Procs. of IJCAI.*

W.W. Cohen, P. Ravikumar, and S.E. Fienberg. 2003. A comparison of string distance metrics for name-matching tasks. In *IIWeb*.

Cleo Condoravdi, Dick Crouch, Valeria de Paiva, Reinhard Stolle, and Daniel G. Bobrow. 2003. Entailment, intensionality and text understanding. In *Proceedings of the HLT-NAACL 2003 workshop on Text meaning*, pages 38–45, Morristown, NJ, USA. Association for Computational Linguistics.

I. Dagan, O. Glickman, and B. Magnini. 2005. The PASCAL Recognising Textual Entailment Challenge. *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, pages 1–8.

Marie-Catherine de Marneffe, Anna Rafferty, and Christopher D. Manning. 2008. Finding contradictions in text. In *ACL 2008*.

A.P. Dempster, N.M. Laird, and D.B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B*, 39(1):1–38.

D. Downey, O. Etzioni, and S. Soderland. 2005. A Probabilistic Model of Redundancy in Information Extraction. In *Procs. of IJCAI.*

Sanda Harabagiu, Andrew Hickl, and Finley Lacatusu. 2006. Negation, contrast and contradiction in text processing. In *AAAI*.

Ykä Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. 1999. TANE: An efficient algorithm for discovering functional and approximate dependencies. *The Computer Journal*, 42(2):100–111.

B. MacCartney and C.D. Manning. 2007. Natural Logic for Textual Inference. In *Workshop on Textual Entailment and Paraphrasing*.

Ellen M. Voorhees. 2008. Contradictions and justifications: Extensions to the textual entailment task. In *Proceedings of ACL-08: HLT*, pages 63–71, Columbus, Ohio, June. Association for Computational Linguistics.

Hong Yao and Howard J. Hamilton. 2008. Mining functional dependencies from data. *Data Min. Knowl. Discov.*, 16(2):197–219.

A. Yates and O. Etzioni. 2007. Unsupervised resolution of objects and relations on the Web. In *Procs. of HLT*.

Regular Expression Learning for Information Extraction

Yunyao Li, Rajasekar Krishnamurthy, Sriram Raghavan, Shivakumar Vaithyanathan
IBM Almaden Research Center
San Jose, CA 95120
`{yunyaoli, rajase, rsriram}@us.ibm.com, shiv@almaden.ibm.com`

H. V. Jagadish[*]
Department of EECS
University of Michigan
Ann Arbor, MI 48109
`jag@umich.edu`

Abstract

Regular expressions have served as the dominant workhorse of practical information extraction for several years. However, there has been little work on reducing the manual effort involved in building high-quality, complex regular expressions for information extraction tasks. In this paper, we propose ReLIE, a novel transformation-based algorithm for learning such complex regular expressions. We evaluate the performance of our algorithm on multiple datasets and compare it against the CRF algorithm. We show that ReLIE, in addition to being an order of magnitude faster, outperforms CRF under conditions of limited training data and cross-domain data. Finally, we show how the accuracy of CRF can be improved by using features extracted by ReLIE.

1 Introduction

A large class of entity extraction tasks can be accomplished by the use of carefully constructed regular expressions (regexes). Examples of entities amenable to such extractions include *email addresses* and *software names* (web collections), *credit card numbers* and *social security numbers* (email compliance), and *gene and protein names* (bioinformatics), etc. These entities share the characteristic that their key representative patterns (features) are expressible in standard constructs of regular expressions. At first glance, it may seem that constructing a regex to extract such entities is fairly straightforward. In reality, robust extraction requires the use of rather complex expressions, as illustrated by the following example.

Example 1 (Phone number extraction)**.** *An obvious pattern for identifying phone numbers is "blocks of digits separated by hyphens" represented as* R_1 = `(\d+\-)+\d+`.[1] *While* R_1 *matches valid phone numbers like* 800-865-1125 *and* 725-1234, *it suffers from both "precision" and "recall" problems. Not only does* R_1 *produce incorrect matches (e.g., social security numbers like* 123-45-6789*), it also fails to identify valid phone numbers such as* 800.865.1125*, and* (800)865-CARE. *An improved regex that addresses these problems is* R_2 = `(\d{3}[-.\ ()]){1,2}[\dA-Z]{4}`.

While multiple machine learning approaches have been proposed for information extraction in recent years (McCallum et al., 2000; Cohen and McCallum, 2003; Klein et al., 2003; Krishnan and Manning, 2006), manually created regexes remain a widely adopted practical solution for information extraction (Appelt and Onyshkevych, 1998; Fukuda et al., 1998; Cunningham, 1999; Tanabe and Wilbur, 2002; Li et al., 2006; DeRose et al., 2007; Zhu et al., 2007). Yet, with a few notable exceptions, which we discuss later in Section 1.1, there has been very little work in reducing this human effort through the use of automatic learning techniques. In this paper, we propose a novel formulation of the problem of learn-

[*]Supported in part by NSF 0438909 and NIH 1-U54-DA021519.

[1]Throughout this paper, we use the syntax of the standard Java regex engine (Java, 2008).

ing regexes for information extraction tasks. We demonstrate that high quality regex extractors can be learned with significantly reduced manual effort. To motivate our approach, we first discuss prior work in the area of learning regexes and describe some of the limitations of these techniques.

1.1 Learning Regular Expressions

The problem of inducing regular languages from positive and negative examples has been studied in the past, even outside the context of information extraction (Alquezar and Sanfeliu, 1994; Dupont, 1996; Firoiu et al., 1998; Garofalakis et al., 2000; Denis, 2001; Denis et al., 2004; Fernau, 2005; Galassi and Giordana, 2005; Bex et al., 2006). Much of this work assumes that the target regex is small and compact thereby allowing the learning algorithm to exploit this information. Consider, for example, the learning of patterns motivated by DNA sequencing applications (Galassi and Giordana, 2005). Here the input sequence is viewed as multiple atomic events separated by gaps. Since each atomic event is easily described by a small and compact regex, the problem reduces to one of learning simple regexes. Similarly, in XML DTD inference (Garofalakis et al., 2000; Bex et al., 2006), it is possible to exploit the fact that the XML documents of interest are often described using simple DTDs. E.g., in an online books store, each book has a title, one or more authors and price. This information can be described in a DTD as $\langle \mathsf{book} \rangle \leftarrow \langle \mathsf{title} \rangle \langle \mathsf{author} \rangle + \langle \mathsf{price} \rangle$. However, as shown in Example 1, regexes for information extraction rely on more complex constructs.

In the context of information extraction, prior work has concentrated primarily on learning regexes over relatively small alphabet sizes. A common theme in (Soderland, 1999; Ciravegna, 2001; Wu and Pottenger, 2005; Feldman et al., 2006) is the problem of learning regexes over tagged tokens produced by other text-processing steps such as POS tagging, morphological analysis, and gazetteer matching. Thus, the alphabet is defined by the space of possible tags output by these analysis steps. A similar approach has been proposed in (Brill, 2000) for POS disambiguation. In contrast, our paper addresses extraction tasks that require "fine-grained" control to accurately capture the structural features of the entity of interest. Consequently, the domain of interest consists of *all characters* thereby dramatically increasing the size of the alphabet. To enable this scale-up, the techniques presented in this paper exploit advanced syntactic constructs (such as character classes and quantifiers) supported by modern regex languages.

Finally, we note that almost all of the above described work define the learning problem over a restricted class of regexes. Typically, the restrictions involve either disallowing or limiting the use of Kleene disclosure and disjunction operations. However, our work imposes no such restrictions.

1.2 Contributions

In a key departure from prior formulations, the learning algorithm presented in this work takes as input not just labeled examples but also an *initial regular expression*. The use of an initial regex has two major advantages. First, this expression provides a natural mechanism for a domain expert to provide domain knowledge about the structure of the entity being extracted. Second, as we show in Section 2, the space of output regular expressions under consideration can be meaningfully restricted by appropriately defining their relationship to the input expression. Such a principled approach to restrict the search space permits the learning algorithm to consider complex regexes in a tractable manner. In contrast, prior work defined a tractable search space by placing restrictions on the target class of regular expressions. Our specific contributions are:

- A novel regex learning problem consisting of learning an "improved" regex given an initial regex and labeled examples
- Formulation of this learning task as an optimization problem over a search space of regexes
- ReLIE, a regex learning algorithm that employs transformations to navigate the search space
- Extensive experimental results over multiple datasets to show the effectiveness of ReLIE and a comparison study with the Conditional Random Field (CRF) algorithm
- Finally, experiments that demonstrate the benefits of using ReLIE as a feature extractor for CRF and possibly other machine learning algorithms.

2 The Regex Learning Problem

Consider the task of identifying instances of some entity $\mathcal{E}$. Let R_0 denote the input regex provided by the user and let $\mathrm{M}(R_0, \mathcal{D})$ denote the set of matches obtained by evaluating R_0 over a document collection $\mathcal{D}$. Let $\mathrm{M_p}(R_0,\mathcal{D}) = \{x \in \mathrm{M}(R_0,\mathcal{D}) : x \text{ instance of } \mathcal{E}\}$ and $\mathrm{M_n}(R_0,\mathcal{D}) = \{x \in \mathrm{M}(R_0,\mathcal{D}) : x \text{ not an instance of } \mathcal{E}\}$ denote the set of positive and negative matches for R_0. Note that a match is positive if it corresponds to an instance of the entity of interest and is negative otherwise. The goal of our learning task is to produce a regex that is "better" than R_0 at identifying instances of $\mathcal{E}$.

Given a candidate regex R, we need a mechanism to judge whether R is indeed a better extractor for $\mathcal{E}$ than R_0. To make this judgment even for just the original document collection $\mathcal{D}$, we must be able to label each instance matched by R (i.e., each element of $\mathrm{M}(R,\mathcal{D})$) as positive or negative. Clearly, this can be accomplished if the set of matches produced by R are contained within the set of available labeled examples, i.e., if $\mathrm{M}(R,\mathcal{D}) \subseteq \mathrm{M}(R_0,\mathcal{D})$. Based on this observation, we make the following assumption:

Assumption 1. *Given an input regex R_0 over some alphabet Σ, any other regex R over Σ is a candidate for our learning algorithm only if* $\mathrm{L}(R) \subseteq \mathrm{L}(R_0)$. *(L(R) denotes the language accepted by R).*

Even with this assumption, we are left with a potentially infinite set of candidate regexes from which our learning algorithm must choose one. To explore this set in a principled fashion, we need a mechanism to move from one element in this space to another, i.e., from one candidate regex to another. In addition, we need an objective function to judge the extraction quality of each candidate regex. We address these two issues below.

Regex Transformations To systematically explore the search space, we introduce the concept of *regex transformations*.

Definition 1 (Regex Transformation). *Let $\mathcal{R}_\Sigma$ denote the set of all regular expressions over some alphabet Σ. A regex transformation is a function $\boldsymbol{T} : \mathcal{R}_\Sigma \to 2^{\mathcal{R}_\Sigma}$ such that $\forall R' \in \boldsymbol{T}(R)$, $\mathrm{L}(R') \subseteq \mathrm{L}(R)$.*

For example, by replacing different occurrences of the quantifier + in R_1 from Example 1 with specific ranges (such as `{1,2}` or `{3}`), we obtain expressions such as R_3 = `(\d+\-){1,2}\d+` and R_4 = `(\d{3}\-)+\d+`. The operation of replacing quantifiers with restricted ranges is an example of a particular class of transformations that we describe further in Section 3. For the present, it is sufficient to view a transformation as a function applied to a regex R that produces, as output, a set of regexes that accept sublanguages of $\mathrm{L}(R)$. We now define the search space of our learning algorithm as follows:

Definition 2 (Search Space). *Given an input regex R_0 and a set of transformations $\mathcal{T}$, the search space of our learning algorithm is $\mathcal{T}(R_0)$, the set of all regexes obtained by (repeatedly) applying the transformations in $\mathcal{T}$ to R_0.*

For instance, if the operation of restricting quantifiers that we described above is part of the transformation set, then R_3 and R_4 are in the search space of our algorithm, given R_1 as input.

Objective Function We now define an objective function, based on the well known F-measure, to compare the extraction quality of different candidate regexes in our search space. Using $\mathrm{M_p}(R,\mathcal{D})$ (resp. $\mathrm{M_n}(R,\mathcal{D})$) to denote the set of positive (resp. negative) matches of a regex R, we define

$$\mathrm{precision}(R,\mathcal{D}) = \frac{\mathrm{M_p}(R,\mathcal{D})}{\mathrm{M_p}(R,\mathcal{D}) + \mathrm{M_n}(R,\mathcal{D})}$$

$$\mathrm{recall}(R,\mathcal{D}) = \frac{\mathrm{M_p}(R,\mathcal{D})}{\mathrm{M_p}(R_0,\mathcal{D})}$$

$$\mathcal{F}(R,\mathcal{D}) = \frac{2 \cdot \mathrm{precision}(R,\mathcal{D}) \cdot \mathrm{recall}(R,\mathcal{D})}{\mathrm{precision}(R,\mathcal{D}) + \mathrm{recall}(R,\mathcal{D})}$$

The regex learning task addressed in this paper can now be formally stated as the following optimization problem:

Definition 3 (**Regex Learning Problem**). *Given an input regex R_0, a document collection $\mathcal{D}$, labeled sets of positive and negative examples $\mathrm{M_p}(R_0,\mathcal{D})$ and $\mathrm{M_n}(R_0,\mathcal{D})$, and a set of transformations $\mathcal{T}$, compute the output regex $R_f = \mathrm{argmax}_{R \in \mathcal{T}(R_0)} \mathcal{F}(R,\mathcal{D})$.*

3 Instantiating Regex Transformations

In this section, we describe how transformations can be implemented by exploiting the syntactic constructs of modern regex engines. To help with our description, we introduce the following task:

Example 2 (Software name extraction). *Consider the task of identifying names of software products in text. A simple pattern for this task is: "one or more capitalized words followed by a version number", represented as R_5 =* `([A-Z]\w*\s*)+[Vv]?(\d+\.?)+`.

When applied to a collection of University web pages, we discovered that R_5 identified correct instances such as *Netscape 2.0*, *Windows 2000* and *Installation Designer v1.1*. However, R_5 also extracted incorrect instances such as course numbers (e.g. *ENGLISH 317*), room numbers (e.g. *Room 330*), and section headings (e.g. *Chapter 2.2*). To eliminate spurious matches such as *ENGLISH 317*, let us enforce the condition that "each word is a single upper-case letter followed by one or more lower-case letters". To accomplish this, we focus on the sub-expression of R_5 that identifies capitalized words, R_{5_1} = `([A-Z]\w*\s*)+`, and replace it with $R_{5_{1a}}$ = `([A-Z][a-z]*\s*)+`. The regex resulting from R_5 by replacing R_{5_1} with $R_{5_{1a}}$ will avoid matches such as *ENGLISH 317*.

An alternate way to improve R_5 is by explicitly disallowing matches against strings like *ENGLISH*, *Room* and *Chapter*. To accomplish this, we can exploit the negative lookahead operator supported in modern regex engines. Lookaheads are special constructs that allow a sequence of characters to be checked for matches against a regex without the characters themselves being part of the match. As an example, `(?!`R_a`)`R_b ("`?!`" being the negative lookahead operator) returns matches of regex R_b but only if they do not match R_a. Thus, by replacing R_{5_1} in our original regex with $R_{5_{1b}}$ =`(?! ENGLISH|Room|Chapter)[A-Z]\w*\s*`, we produce an improved regex for software names.

The above examples illustrate the general principle of our transformation technique. In essence, we isolate a sub-expression of a given regex R and modify it such that the resulting regex accepts a sub-language of R. We consider two kinds of modifications – drop-disjunct and include-intersect. In drop-disjunct, we operate on a sub-expression that corresponds to a disjunct and drop one or more operands of that disjunct. In include-intersect, we restrict the chosen sub-expression by intersecting it with some other regex. Formally,

Definition 4 (Drop-disjunct Transformation). *Let $R \in \mathcal{R}_\Sigma$ be a regex of the form $R = R_a\rho(X)R_b$, where $\rho(X)$ denotes the disjunction $R_1|R_2|\ldots|R_n$ of any non-empty set of regexes $X = \{R_1, R_2, \ldots, R_n\}$. The drop-disjunct transformation $\mathrm{DD}(R, X, Y)$ for some $Y \subset X$, $Y \neq \emptyset$ results in the new regex $R_a\rho(Y)R_b$.*

Definition 5 (Include-Intersect Transformation). *Let $R \in \mathcal{R}_\Sigma$ be a regex of the form $R = R_aXR_b$ for some $X \in \mathcal{R}_\Sigma$, $X \neq \emptyset$. The include-intersect transformation $\mathrm{II}(R, X, Y)$ for some $Y \in \mathcal{R}_\Sigma$, $Y \neq \emptyset$ results in the new regex $R_a(X \cap Y)R_b$.*

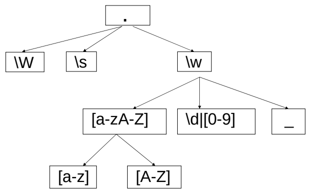


Figure 1: Sample Character Classes in Regex

We state the following proposition (proof omitted in the interest of space) that guarantees that both drop-disjunct and include-intersect restrict the language of the resulting regex, and therefore are valid transformations according to Definition 1.

Proposition 1. *Given regexes R, X_1, Y_1, X_2 and Y_2 from $\mathcal{R}_\Sigma$ such that $\mathrm{DD}(R, X_1, Y_1)$ and $\mathrm{II}(R, X_2, Y_2)$ are applicable, $\mathrm{L}(\mathrm{DD}(R, X_1, Y_1)) \subseteq \mathrm{L}(R)$ and $\mathrm{L}(\mathrm{II}(R, X_2, Y_2)) \subseteq \mathrm{L}(R)$.*

We now proceed to describe how we use different syntactic constructs to apply drop-disjunct and include-intersect transformations.

Character Class Restrictions Character classes are short-hand notations for denoting the disjunction of a set of characters (`\d` is equivalent to `(0|1...|9)`; `\w` is equivalent to `(a|...|z|A|...|Z|0|1...|9|_)`; etc.).[2] Figure 1 illustrates a character class hierarchy in which each node is a stricter class than its parent (e.g., `\d` is stricter than `\w`). A replacement of any of these character classes by one of its descendants is an instance of the *drop-disjunct* transformation. Notice that in Example 2, when replacing R_{5_1} with $R_{5_{1a}}$, we were in effect applying a character class restriction.

Quantifier Restrictions Quantifiers are used to define the range of valid counts of a repetitive sequence. For instance, `a{m,n}` looks for a sequence of `a`'s of length at least `m` and at most `n`. Since quantifiers are also disjuncts (e.g., `a{1,3}` is equivalent to `a|aa|aaa`), the replacement of an expression $R\{m, n\}$ with an expression $R\{m_1, n_1\}$ ($m \leq m_1 \leq n_1 \leq n$) is an instance of the *drop-disjunct* transformation. For example, given a subexpression of the form `a{1,3}`, we can replace it with

[2]Note that there are two distinct character classes `\W` and `\w`

one of `a{1,1}`, `a{1,2}`, `a{2,2}`, `a{2,3}`, or `a{3,3}`. Note that, before applying this transformation, wildcard expressions such as `a+` and `a*` are replaced by `a{0,maxCount}` and `a{1,maxCount}` respectively, where `maxCount` is a user configured maximum length for the entity being extracted.

Negative Dictionaries Observe that the include-intersect transformation (Definition 5) is applicable for every possible sub-expression of a given regex R. Note that a valid *sub-expression* in R is any portion of R where a capturing group can be introduced.[3] Consider a regex $R = R_a X R_b$ with a sub-expression X; the application of *include-intersect* requires another regex Y to yield $R_a(X \cap Y)R_b$. We would like to construct Y such that $R_a(X \cap Y)R_b$ is "better" than R for the task at hand. Therefore, we construct Y as $\neg Y'$ where Y' is a regex constructed from negative matches of R. Specifically, we look at each negative match of R and identify the substring of the match that corresponds to X. We then apply a greedy heuristic (see below) to these substrings to yield a *negative dictionary* Y'. Finally, the transformed regex $R_a(X \cap \neg Y')R_b$ is implemented using the negative lookahead expression R_a`(?! Y')X`R_b.

Greedy Heuristic for Negative Dictionaries Implementation of the above procedure requires certain judicious choices in the construction of the negative dictionary to ensure tractability of this transformation. Let $S(X)$ denote the distinct strings that correspond to the sub-expression X in the negative matches of R.[4] Since any subset of $S(X)$ is a candidate negative dictionary, we are left with an exponential number of possible transformations. In our implementation, we used a greedy heuristic to pick a single negative dictionary consisting of all those elements of $S(X)$ that *individually* improve the F-measure. For instance, in Example 2, if the independent substitution of R_{5_1} with `(?!ENGLISH)[A-Z]\w*\s*`, `(?!Room)[A-Z]\w*\s*`, and `(?!Chapter)[A-Z]\w*\s*` each improves the F-measure, we produce a negative dictionary consisting of `ENGLISH`, `Room`, and `Chapter`. This is precisely how the disjunct `ENGLISH|Room|Chapter` is constructed in $R_{5_{1b}}$.

[3]For instance, the sub-expressions of `ab{1,2}c` are `a`, `ab{1,2}`, `ab{1,2}c`, `b`, `b{1,2}`, `b{1,2}c`, and `c`.

[4]S(X) can be obtained automatically by identifying the substring corresponding to the group X in each entry in $\mathrm{M_n}(R,\mathcal{D})$

```
Procedure ReLIE($\mathcal{M}_{tr}$,$\mathcal{M}_{val}$,$R_0$,$\mathcal{T}$)
// $\mathcal{M}_{tr}$: set of labeled matches used as training data
// $\mathcal{M}_{val}$: set of labeled matches used as validation data
// $R_0$: user-provided regular expression
// $\mathcal{T}$: set of transformations
begin
 1. $R_{new} = R_0$
 2. do {
 3.     for each transformation $t_i \in \mathcal{T}$
 4.         Candidate$_i$=ApplyTransformations($R_{new}, t_i$)
 5.     let Candidates = $\bigcup_i$ Candidate$_i$
 6.     let $R'$ = $\text{argmax}_{R \in \text{Candidates}} \mathcal{F}(R, \mathcal{M}_{tr})$
 7.     if ($\mathcal{F}(R', \mathcal{M}_{tr}) <= \mathcal{F}(R_{new}, \mathcal{M}_{tr})$) return $R_{new}$
 8.     if ($\mathcal{F}(R', \mathcal{M}_{val}) < \mathcal{F}(R_{new}, \mathcal{M}_{val})$) return $R_{new}$
 9.     $R_{new}$ = $R'$
10. } while(true)
end
```

Figure 2: ReLIE Search Algorithm

4 ReLIE Search Algorithm

Figure 2 describes the ReLIE algorithm for the Regex Learning Problem (Definition 3) based on the transformations described in Section 3. ReLIE is a greedy hill climbing search procedure that chooses, at every iteration, the regex with the highest F-measure. An iteration in ReLIE consists of:

- Applying every transformation on the current regex R_{new} to obtain a set of candidate regexes
- From the candidates, choosing the regex R' whose F-measure over the training dataset is maximum

To avoid overfitting, ReLIE terminates when either of the following conditions is true: (i) there is no improvement in F-measure over the training set; (ii) there is a drop in F-measure when applying R' on the validation set.

The following proposition provides an upper bound for the running time of the ReLIE algorithm.

Proposition 2. *Given any valid set of inputs $\mathcal{M}_{tr}$, $\mathcal{M}_{val}$, R_0, and $\mathcal{T}$, the ReLIE algorithm terminates in at most $|\,\mathrm{M_n}(R_0, \mathcal{M}_{tr})|$ iterations. The running time of the algorithm $T_{Total}(R_0, \mathcal{M}_{tr}, \mathcal{M}_{val}) \leq |\,\mathrm{M_n}(R_0, \mathcal{M}_{tr})| * t_0$, where t_0 is the time taken for the first iteration of the algorithm.*

Proof. With reference to Figure 2, in each iteration, the F-measure of the "best" regex R' is strictly better than R_{new}. Since $L(R') \subseteq L(R_{new})$, R' eliminates at least one additional negative match compared to R_{new}. Hence, the maximum number of iterations is $|\,\mathrm{M_n}(R_0, \mathcal{M}_{tr})|$.

For a regular expression R, let $n_{cc}(R)$ and $n_q(R)$ denote, respectively, the number of character classes and quantifiers in R. The maximum number of possible sub-expressions in R is $|R|^2$, where $|R|$ is the length of R. Let $MaxQ(R)$ denote the maximum number of ways in which a single quantifier appearing in R can be restricted to a smaller range. Let F_{cc} denote the maximum fanout[5] of the character class hierarchy. Let $T_{ReEval}(\mathcal{D})$ denote the average time taken to evaluate a regex over dataset $\mathcal{D}$.

Let R_i denote the regex at the beginning of iteration i. The number of candidate regexes obtained by applying the three transformations is

$$\textit{NumRE}(R_i, \mathcal{M}_{tr}) \leq n_{cc}(R_i) * F_{cc} + n_q(R_i) * \textit{MaxQ}(R_i) + |R_i|^2$$

The time taken to enumerate the character class and quantifier restriction transformations is proportional to the resulting number of candidate regexes. The time taken for the negative dictionaries transformation is given by the running time of the greedy heuristic (Section 3). The total time taken to enumerate all candidate regexes is given by (for some constant c)

$$T_{Enum}(R_i, \mathcal{M}_{tr}) \leq c * (n_{cc}(R_i) * F_{cc} + n_q(R_i) * \textit{MaxQ}(R_i) + |R_i|^2 * \mathrm{M_n}(R_i, \mathcal{M}_{tr}) * T_{ReEval}(\mathcal{M}_{tr}))$$

Choosing the best transformation involves evaluating each candidate regex over the training and validation corpus and the time taken for this step is

$$T_{PickBest}(R_i, \mathcal{M}_{tr}, \mathcal{M}_{val}) = \textit{NumRE}(R_i, \mathcal{M}_{tr}) * (T_{ReEval}(\mathcal{M}_{tr}) + T_{ReEval}(\mathcal{M}_{val}))$$

The total time taken for an iteration can be written as

$$T_I(R_i, \mathcal{M}_{tr}, \mathcal{M}_{val}) = T_{Enum}(R_i, \mathcal{M}_{tr}) + T_{PickBest}(R_i, \mathcal{M}_{tr}, \mathcal{M}_{val})$$

It can be shown that the time taken in each iteration decreases monotonically (details omitted in the interest of space). Therefore, the total running time of the algorithm is given by

$$T_{Total}(R_0, \mathcal{M}_{tr}, \mathcal{M}_{val}) = \sum T_I(R_i, \mathcal{M}_{tr}, \mathcal{M}_{val}) \leq |\mathrm{M_n}(R_0, \mathcal{M}_{tr})| * t_0.$$

where $t_0 = T_I(R_0, \mathcal{M}_{tr}, \mathcal{M}_{val})$ is the running time of the first iteration of the algorithm. $\square$

5 Experiments

In this section, we present an empirical study of the ReLIE algorithm using four extraction tasks over three real-life data sets. The goal of this study is to evaluate the effectiveness of ReLIE in learning complex regexes and to investigate how it compares with standard machine learning algorithms.

5.1 Experimental Setup

Data Set The datasets used in our experiments are:

- **`EWeb:`** A collection of 50,000 web pages crawled from a corporate intranet.
- **`AWeb:`** A set of 50,000 web pages obtained from the publicly available University of Michigan Web page collection (Li et al., 2006), including a subcollection of 10,000 pages (**`AWeb-S`**).
- **`Email:`** A collection of 10,000 emails obtained from the publicly available Enron email collection (Minkov et al., 2005).

Extraction Tasks *SoftwareNameTask*, *CourseNumberTask* and *PhoneNumberTask* were evaluated on `EWeb`, `AWeb` and `Email`, respectively. Since web pages have large number of URLs, to keep the labeling task manageable, *URLTask* was evaluated on `AWeb-S`.

Gold Standard For each task, the gold standard was created by manually labeling all matches for the initial regex. Note that only exact matches with the gold standard are considered correct in our evaluations. [6]

Comparison Study To evaluate ReLIE for entity extraction vis-a-vis existing algorithms, we used the popular conditional random field (CRF). Specifically, we used the MinorThird (Cohen, 2004) implementation of CRF to train models for all four extraction tasks. For training the CRF we provided it with the set of positive and negative matches from the initial regex with a context of 200 characters on either side of each match[7]. Since it is unlikely that useful features are located far away from the entity, we believe that 200 characters on either side is sufficient context. The CRF used the base features described in (Cohen et al., 2005). To ensure fair comparison with ReLIE, we also included the matches corresponding to the input regex as a feature to the CRF. In practice, more complex features (e.g., dictionaries, simple regexes) derived by domain experts are often provided to CRFs. However, such features can also be used to refine the initial regex given to ReLIE. Hence, with a view to investigating the "raw" learning capability of the two approaches, we chose to run all our experiments without any additional manually derived features. In fact, the patterns learned by ReLIE through transformations are often similar

[5]Fanout is the number of ways in which a character class may be restricted as defined by the hierarchy (e.g. Figure 1).

[6]The labeled data will be made publicly available at `http://www.eecs.umich.edu/db/regexLearning/`.

[7]Ideally, we would have preferred to let MinorThird extract appropriate features from complete documents in the training-set but could not get it to load our large datasets.

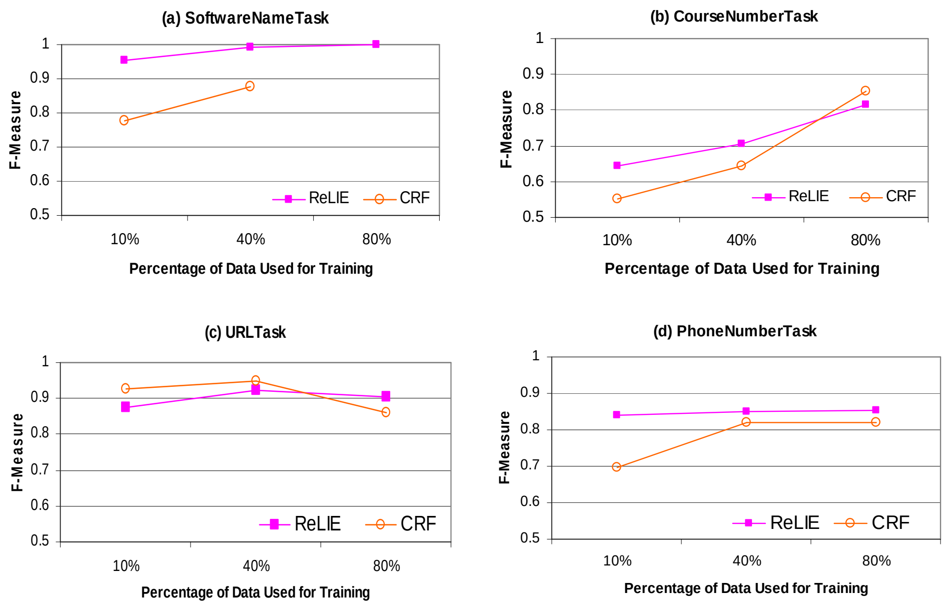


Figure 3: Extraction Quality[a]

[a]For *SoftwareNameTask*, with 80% training data we could not obtain results for CRF as the program failed repeatedly during the training phase.

to the features that domain experts may provide to CRF. We will revisit this issue in Section 5.4.

Evaluation We used the standard F-measure to evaluate the effectiveness of ReLIE and CRF. We divided each dataset into 10 equal parts and used X% of the dataset for training (X=10, 40 and 80), 10% for validation, and remaining (90-X)% for testing. All results are reported on the test set.

5.2 Results

Four extraction tasks were chosen to reflect the entities commonly present in the three datasets.

- *SoftwareNameTask*: Extracting software names such as Lotus Notes 8.0, Open Office Suite 2007.
- *CourseNumberTask*: Extracting university course numbers such as EECS 584, Pharm 101.
- *PhoneNumberTask*: Extracting phone numbers such as 1-800-COMCAST, (425)123 5678.
- *URLTask*: Extracting URLs such as http:\\www.abc.com and lsa.umich.edu/ foo/.[8]

This section summarizes the results of our empirical evaluation comparing ReLIE and CRF.

Raw Extraction Quality The cross-validated results across all four tasks are presented in Figure 3.

- With 10% training data, ReLIE outperforms CRF on three out of four tasks with a difference in F-measure ranging from 0.1 to 0.2.
- As training data increases, both algorithms perform better with the gap between the two reducing for all the four tasks. For *CourseNumberTask* and *URLTask*, CRF does slightly better than ReLIE for larger training dataset. For the other two tasks, ReLIE retains its advantage over CRF.[9]

The above results indicate that ReLIE performs comparably with CRF with a slight edge in conditions of limited training data. Indeed, the capability to learn high-quality extractors using a small training set is important because labeled data is often expensive to obtain. For precisely this same reason, we would ideally like to learn the extractors once and then apply them to other datasets as needed. Since these other datasets may be from a different domain, we next performed a cross-domain test (i.e., training

[8]*URLTask* may appear to be simplistic. However, extracting URLs without the leading protocol definitions (e.g. `http`) can be challenging.

[9]For *SoftwareNameTask*, with 80% training data we could not obtain results for CRF as the program failed repeatedly during the training phase.

and testing on different domains).

Cross-domain Evaluation Table 1 summarizes the results of training the algorithms on one data set and testing on another. The scenarios chosen are: (i) *SoftwareNameTask* trained on `EWeb` and tested on `AWeb`, (ii) *URLTask* trained on `AWeb` and tested on `Email`, and (iii) *PhoneNumberTask* trained on `Email` and tested on `AWeb`.[10] We can see that ReLIE significantly outperforms CRF for all three tasks, even when provided with a large training dataset. Compared to testing on the same dataset, there is a reduction in F-measure (less than 0.1 in many cases) when the regex learned by ReLIE is applied to a different dataset, while the drop for CRF is much more significant (over 0.5 in many cases).[11]

Task(Training, Testing) \ Data for Training	10%		40%		80%	
	ReLIE	CRF	ReLIE	CRF	ReLIE	CRF
SoftwareNameTask(`EWeb`,`AWeb`)	**0.920**	0.297	**0.977**	0.503	**0.971**	N/A
URLTask(`AWeb-S`,`Email`)	**0.690**	0.209	**0.784**	0.380	**0.801**	0.507
PhoneNumberTask(`Email`,`AWeb`)	**0.357**	0.130	**0.475**	0.125	**0.513**	0.120

Table 1: Cross Domain Test (F-measure).

Technique	*SoftwareNameTask*		*CourseNumberTask*		*URLTask*		*PhoneNumberTask*	
	training	testing	training	testing	training	testing	training	testing
ReLIE	511.7	20.6	69.3	18.4	73.8	7.7	39.4	1.1
CRF	7597.0	2315.8	482.5	75.4	438.7	53.8	434.8	57.7
$\frac{t(\text{ReLIE})}{t(\text{CRF})}$	0.067	0.009	0.144	0.244	0.168	0.143	0.091	0.019

Table 2: Average Training/Testing Time (sec)(with 40% data for training)

Task(Extra Feature) \ Data for Training	10%		40%		80%	
	CRF	C+RL	CRF	C+RL	CRF	C+RL
CourseNumberTask(Negative Dictionary)	0.553	**0.624**	0.644	**0.764**	**0.854**	0.845
PhoneNumberTask(Quantifier)	0.695	**0.893**	0.820	**0.937**	0.821	**0.964**

Table 3: ReLIE as Feature Extractor (C+RL is CRF enhanced with features learned by ReLIE).

Training Time Another issue of practical consideration is the efficiency of the learning algorithm. Table 2 reports the average training and testing time for both algorithms on the four tasks. On average ReLIE is *an order of magnitude faster* than CRF in both building the model and applying the learnt model.

Robustness to Variations in Input Regexes The transformations done by ReLIE are based on the structure of the input regex. Therefore given different input regexes, the final regexes learned by ReLIE will be different. To evaluate the impact of the structure of the input regex on the quality of the regex learned by ReLIE, we started with different regexes[12]

for the same task. We found that ReLIE is robust to variations in input regexes. For instance, on *SoftwareNameTask*, the standard deviation in F-measure of the final regexes generated from six different input regexes was less than 0.05. Further details of this experiment are omitted in the interest of space.

5.3 Discussion

The results of our comparison study (Figure 3) indicates that for raw extraction quality ReLIE has a slight edge over CRF for small training data. However, in cross-domain performance (Table 1) ReLIE is significantly better than CRF (by 0.41 on average) . To understand this discrepancy, we examined the final regex learned by ReLIE and compared that with the features learned by CRF. Examples of initial regexes with corresponding final regexes learnt by ReLIE with 10% training data are listed in Table 4. Recall, from Section 3, that ReLIE transformations include character class restrictions, quantifier restrictions and addition of negative dictionaries. For instance, in the *SoftwareNameTask*, the final regex listed was obtained by restricting `[a-zA-Z]` to `[a-z]`, `\w` to `[a-zA-Z]`, and adding the negative dictionary `(Copyright|Fall|···|Issue)`. Similarly, for the *PhoneNumberTask*, the final regex involved two negative dictionaries (expressed as `(?![,])` and `(?![,:])`) [13] and quantifier restrictions (e.g. the first `[A-Z\d]{2,4}` was transformed

[10]We do not report results for *CourseNumberTask* as course numbers are specific to academic webpages and do not appear in the other two domains

[11]Similar cross-domain performance deterioration for a machine learning approach has been observed by (Guo et al., 2006).

[12]Recall that the search space of ReLIE is limited by $L(R_0)$ (Assumption 1). Thus to ensure meaningful comparison, for the same task any two given input regexes R_0 and R'_0 are chosen in such a way that although their structures are different, $\mathrm{M_p}(R_0, \mathcal{D}) = \mathrm{M_p}(R'_0, \mathcal{D})$ and $\mathrm{M_n}(R_0, \mathcal{D}) = \mathrm{M_n}(R'_0, \mathcal{D})$.

[13]To obtain these negative dictionaries, ReLIE not only needs to correctly identify the dictionary entries from negative matches but also has to place the corresponding negative lookahead expression at the appropriate place in the regex.

SoftwareNameTask	R_0	`\b([A-Z][a-zA-Z]{1,10}\s){1,5}\s*(\w{0,2}\d[\.]?){1,4}\b`
	R_{final}	`\b((?!(Copyright|Page|Physics|Question|···|Article|Issue))[A-Z][a-z]{1,10}\s){1,5}\s*([a-zA-Z]{0,2}\d[\.]?){1,4}\b`
PhoneNumberTask	R_0	`\b(1\W+)?\W?\d{3,3}\W*\s*\W?[A-Z\d]{2,4}\s*\W?[A-Z\d]{2,4}\b`
	R_{final}	`\b(1\W+)?\W?\d{3,3}((?![,])\W*)\s*\W?[A-Z\d]{3,3}\s*((?![,:])\W?)[A-Z\d]{3,4}\b`
CourseNumberTask	R_0	`\b([A-Z][a-zA-Z]+)\s+\d{3,3}\b`
	R_{final}	`\b(((?!(At|Between|···Contact|Some|Suite|Volume))[A-Z][a-zA-Z]+))\s+\d{3,3}\b`
URLTask	R_0	`\b(\w+://)?(\w+\.){0,2}\w+\.\w+(/[^\s]+){0,20}\b`
	R_{final}	`\b((?!(Response_20010702_1607.csv|···))((\w+://)?(\w+\.){0,2}\w+\.(?!(ppt|···doc))[a-zA-Z]{2,3}))(/[^\s]+){0,20}\b`

Table 4: Sample Regular Expressions Learned by ReLIE(R_0: input regex; R_{final}: final regex learned; the parts of R_0 modified by ReLIE and the corresponding parts in R_{final} are highlighted.)

into `[A-Z\d]{3,3}`).

After examining the features learnt by CRF, it was clear that while CRF could learn features such as the negative dictionary it is unable to learn character-level features. This should not be surprising since our CRF was trained with primarily tokens as features (cf. Section 5.1). While this limitation was less of a factor in experiments involving data from the same domain (some effects were seen with smaller training data), it does explain the significant difference between the two algorithms in cross-domain tasks where the vocabulary can be significantly different. Indeed, in practical usage of CRF, the main challenge is to come up with additional complex features (often in the form of dictionary and regex patterns) that need to be given to the CRF (Minkov et al., 2005). Such complex features are largely hand-crafted and thus expensive to obtain. Since the ReLIE transformations are operations over characters, a natural question to ask is: "Can the regex learned by ReLIE be used to provide features to CRF?" We answer this question below.

5.4 ReLIE as Feature Extractor for CRF

To understand the effect of incorporating ReLIE-identified features into CRF, we chose the two tasks (*CourseNumberTask* and *PhoneNumberTask*) with the least F-measure in our experiments to determine raw extraction quality. We examined the final regex produced by ReLIE and manually extracted portions to serve as features. For example, the negative dictionary learned by ReLIE for the *CourseNumberTask* `(At|Between|···|Volume)` was incorporated as a feature into CRF. To help isolate the effects, for each task, we only incorporated features corresponding to a single transformation: negative dictionaries for *CourseNumberTask* and quantifier restrictions for *PhoneNumberTask*. The results of these experiments are shown in Table 3. The first point worthy of note is that performance has improved in all but one case. Second, despite the F-measure on *CourseNumberTask* being lower than *PhoneNumberTask* (presumably more potential for improvement), the improvements on *PhoneNumberTask* are significantly higher. This observation is consistent with our conjecture in Section 5.1 that CRF learns token-level features; therefore incorporating negative dictionaries as extra feature provides only limited improvement. Admittedly more experiments are needed to understand the full impact of incorporating ReLIE-identified features into CRF. However, we do believe that this is an exciting direction of future research.

6 Summary and Future Work

We proposed a novel formulation of the problem of learning complex character-level regexes for entity extraction tasks. We introduced the concept of regex transformations and described how these could be realized using the syntactic constructs of modern regex languages. We presented ReLIE, a powerful regex learning algorithm that exploits these ideas. Our experiments demonstrate that ReLIE is very effective for certain classes of entity extraction, particularly under conditions of cross-domain and limited training data. Our preliminary results also indicate the possibility of using ReLIE as a powerful feature extractor for CRF and other machine learning algorithms. Further investigation of this aspect of ReLIE presents an interesting avenue of future work.

Acknowledgments

We thank the anonymous reviewers for their insightful and constructive comments and suggestions. We are also grateful for comments from David Gondek and Sebastian Blohm.

References

R. Alquezar and A. Sanfeliu. 1994. Incremental grammatical inference from positive and negative data using unbiased finite state automata. In *SSPR*.

Douglas E. Appelt and Boyan Onyshkevych. 1998. The common pattern specification language. In *TIPSTER TEXT PROGRAM*.

Geert Jan Bex et al. 2006. Inference of concise DTDs from XML data. In *VLDB*.

Eric Brill. 2000. Pattern-based disambiguation for natural language processing. In *SIGDAT*.

William W. Cohen and Andrew McCallum. 2003. Information Extraction from the World Wide Web. in *KDD*

William W. Cohen. 2004. Minorthird: Methods for identifying names and ontological relations in text using heuristics for inducing regularities from data. http://minorthird.sourceforge.net.

William W. Cohen et al. 2005. Learning to Understand Web Site Update Requests. In *IJCAI*.

Fabio Ciravegna. 2001. Adaptive information extraction from text by rule induction and generalization. In *IJCAI*.

H. Cunningham. 1999. JAPE – a java annotation patterns engine.

Francois Denis et al. 2004. Learning regular languages using RFSAs. *Theor. Comput. Sci.*, 313(2):267–294.

Francois Denis. 2001. Learning regular languages from simple positive examples. *Machine Learning*, 44(1/2):37–66.

Pedro DeRose et al. 2007. DBLife: A Community Information Management Platform for the Database Research Community In *CIDR*

Pierre Dupont. 1996. Incremental regular inference. In *ICGI*.

Ronen Feldman et all. 2006. Self-supervised Relation Extraction from the Web. In *ISMIS*.

Henning Fernau. 2005. Algorithms for learning regular expressions. In *ALT*.

Laura Firoiu et al. 1998. Learning regular languages from positive evidence. In *CogSci*.

K. Fukuda et al. 1998. Toward information extraction: identifying protein names from biological papers. *Pac Symp Biocomput.*, 1998:707–718

Ugo Galassi and Attilio Giordana. 2005. Learning regular expressions from noisy sequences. In *SARA*.

Minos Garofalakis et al. 2000. XTRACT: a system for extracting document type descriptors from XML documents. In *SIGMOD*.

Hong Lei Guo et al. 2006. Empirical Study on the Performance Stability of Named Entity Recognition Model across Domains In *EMNLP*.

Java Regular Expressions. 2008. http://java.sun.com /javase/6/docs/api/java/util/regex/package-summary.html.

Dan Klein et al. 2003. Named Entity Recognition with Character-Level Models. In *HLT-NAACL*.

Vijay Krishnan and Christopher D. Manning. 2006. An Effective Two-Stage Model for Exploiting Non-Local Dependencies in Named Entity Recognition. In *ACL*.

Yunyao Li et al. 2006. Getting work done on the web: Supporting transactional queries. In *SIGIR*.

Andrew McCallum et al. 2000. Maximum Entropy Markov Models for Information Extraction and Segmentation. In *ICML*.

Einat Minkov et al. 2005. Extracting personal names from emails: Applying named entity recognition to informal text. In *HLT/EMNLP*.

Stephen Soderland. 1999. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34:233–272.

Lorraine Tanabe and W. John Wilbur 2002. Tagging gene and protein names in biomedical text. *Bioinformatics*, 18:1124–1132.

Tianhao Wu and William M. Pottenger. 2005. A semi-supervised active learning algorithm for information extraction from textual data. *JASIST*, 56(3):258–271.

Huaiyu Zhu, Alexander Loeser, Sriram Raghavan, Shivakumar Vaithyanathan 2007. Navigating the intranet with high precision. In *WWW*.

Modeling Annotators: A Generative Approach to Learning from Annotator Rationales[*]

Omar F. Zaidan and **Jason Eisner**
Dept. of Computer Science, Johns Hopkins University
Baltimore, MD 21218, USA
{ozaidan,jason}@cs.jhu.edu

Abstract

A human annotator can provide hints to a machine learner by highlighting contextual "rationales" for each of his or her annotations (Zaidan et al., 2007). How can one exploit this side information to better learn the desired parameters θ? We present a generative model of how a given annotator, knowing the true θ, stochastically chooses rationales. Thus, observing the rationales helps us infer the true θ. We collect substring rationales for a sentiment classification task (Pang and Lee, 2004) and use them to obtain significant accuracy improvements for each annotator. Our new generative approach exploits the rationales more effectively than our previous "masking SVM" approach. It is also more principled, and could be adapted to help learn other kinds of probabilistic classifiers for quite different tasks.

1 Background

Many recent papers aim to reduce the amount of annotated data needed to train the parameters of a statistical model. Well-known paradigms include active learning, semi-supervised learning, and either domain adaptation or cross-lingual transfer from existing annotated data.

A rather different paradigm is to change the actual *task* that is given to annotators, giving them a greater hand in shaping the learned classifier. After all, human annotators themselves are more than just black-box classifiers to be run on training data. They possess some introspective knowledge about their own classification procedure. The hope is to mine this knowledge rapidly via appropriate questions and use it to help train a machine classifier. *How* to do this, however, is still being explored.

1.1 Hand-crafted rules

An obvious option is to have the annotators directly express their knowledge by hand-crafting rules. This approach remains "data-driven" if the annotators repeatedly refine their system against a corpus of labeled or unlabeled examples. This achieves high performance in some domains, such as NP chunking (Brill and Ngai, 1999), but requires more analytical skill from the annotators. One empirical study (Ngai and Yarowsky, 2000) found that it also required more annotation time than active learning.

1.2 Feature selection by humans

More recent work has focused on statistical classifiers. Training such classifiers faces the "credit assignment problem." Given a training example x with many features, which features are responsible for its annotated class y? It may take many training examples to distinguish useful vs. irrelevant features.[1]

To reduce the number of training examples needed, one can ask annotators to examine or propose some candidate features. This is possible even for the very large feature sets that are typically used in NLP. In document classification, Raghavan et al. (2006) show that feature selection by an oracle could be helpful, and that humans are both rapid and reasonably good at distinguishing highly useful n-gram features from randomly chosen ones, even when viewing these n-grams out of context.

Druck et al. (2008) show annotators some features f from a fixed feature set, and ask them to choose a class label y such that $p(y \mid f)$ is as high as possible. Haghighi and Klein (2006) do the reverse: for each class label y, they ask the annotators to propose a few "prototypical" features f such that $p(y \mid f)$ is as high as possible.

1.3 Feature selection in context

The above methods consider features out of context. An annotator might have an easier time examining

[*]This work was supported by National Science Foundation grant No. 0347822 and the JHU WSE/APL Partnership Fund. Special thanks to Christine Piatko for many useful discussions.

[1]Most NLP systems use thousands or millions of features, because it is helpful to include lexical features over a large vocabulary, often conjoined with lexical or non-lexical context.

features *in context* to recognize whether they appear relevant. This is particularly true for features that are only modestly or only sometimes helpful, which may be abundant in NLP tasks.

Thus, Raghavan et al. (2006) propose an active learning method in which, while classifying a training document, the annotator also identifies some features of *that* document as particularly relevant. E.g., the annotator might highlight particular unigrams as he or she reads the document. In their proposal, a feature that is highlighted in any document is assumed to be globally more relevant. Its dimension in feature space is scaled by a factor of 10 so that this feature has more influence on distances or inner products, and hence on the learned classifier.

1.4 Concerns about marking features

Despite the success of the above work, we have several concerns about asking annotators to identify globally relevant features.

First, a feature in isolation really does not have a well-defined worth. A feature may be useful only in conjunction with other features,[2] or be useful only to the extent that other correlated features are *not* selected to do the same work.

Second, it is not clear how an annotator would easily view and highlight features in context, except for the simplest feature sets. In the phrase Apple shares up 3%, there may be several features that fire on the substring Apple—responding to the string Apple, its case-invariant form apple, its lemma apple- (which would also respond to apples), its context-dependent sense Apple_2, its part of speech noun, etc. How does the annotator indicate which of these features are relevant?

Third, annotating features is only appropriate when the feature set can be easily understood by a human. This is not always the case. It would be hard for annotators to read, write, or evaluate a description of a complex syntactic configuration in NLP or a convolution filter in machine vision.

Fourth, traditional annotation efforts usually try to remain agnostic about the machine learning methods and features to be used. The project's cost is justified by saying that the annotations will be reused by many researchers (perhaps in a "shared task"), who are free to compete on how they tackle the learning problem. Unfortunately, feature annotation commits to a particular feature set at annotation time. Subsequent research cannot easily adjust the definition of the features, or obtain annotation of new features.

2 Annotating Rationales

To solve these problems, we propose that annotators should not *select features* but rather *mark relevant portions of the example*. In earlier work (Zaidan et al., 2007), we called these markings "rationales."

For example, when classifying a movie review as positive or negative, the annotator would also highlight phrases that supported that judgment. Figure 1 shows two such rationales.

A multi-annotator timing study (Zaidan et al., 2007) found that highlighting rationale phrases while reading movie reviews only doubled annotation time, although annotators marked 5–11 rationale substrings in addition to the simple binary class. The benefit justified the extra time. Furthermore, much of the benefit could have been obtained by giving rationales for only a fraction of the reviews.

In the visual domain, when classifying an image as containing a zoo, the annotator might circle some animals or cages and the sign reading "Zoo." The Peekaboom *game* (von Ahn et al., 2006) was in fact built to elicit such approximate yet relevant regions of images. Further scenarios were discussed in (Zaidan et al., 2007): rationale annotation for named entities, linguistic relations, or handwritten digits.

Annotating rationales does not require the annotator to think about the feature space, nor even to know anything about it. Arguably this makes annotation easier and more flexible. It also preserves the reusability of the annotated data. Anyone is free to reuse our collected rationales (section 4) to aid in learning a classifier with richer features, or a different kind of classifier altogether, using either our procedures or novel procedures.

3 Modeling Rationale Annotations

As rationales are more indirect than explicit features, they present a trickier machine learning problem.

[2]For example, a linear classifier can learn that most training examples satisfy $A \rightarrow B$ by setting $\theta_A = -5$ and $\theta_{A \wedge B} = +5$, but this solution requires selecting both A and $A \wedge B$ as features. More simply, a polynomial kernel can consider the conjunction $A \wedge B$ only if both A and B are selected as features.

We wish to learn the parameters θ of some classifier. How can the annotator's rationales help us to do this without many training examples? We will have to exploit a presumed relationship between the rationales and the optimal value of θ (i.e., the value that we would learn on an infinite training set).

This paper exploits an explicit, parametric model of that relationship. The model's parameters ϕ are intended to capture what that annotator is doing when he or she marks rationales. Most importantly, they capture how he or she is influenced by the true θ. Given this, our learning method will prefer values of θ that would adequately explain the rationales (as well as the training classifications).

3.1 A generative approach

For concreteness, we will assume that the task is document classification. Our training data consists of n triples $\{(x_1, y_1, r_1), ..., (x_n, y_n, r_n)\}$), where x_i is a document, y_i is its annotated class, and r_i is its rationale markup. At test time we will have to predict y_{n+1} from x_{n+1}, without any r_{n+1}.

We propose to jointly choose parameter vectors θ and ϕ to maximize the following regularized conditional likelihood:[3]

$$\prod_{i=1}^{n} p(y_i, r_i \mid x_i, \theta, \phi) \cdot p_{\text{prior}}(\theta, \phi) \tag{1}$$
$$\overset{\text{def}}{=} \prod_{i=1}^{n} p_\theta(y_i \mid x_i) \cdot p_\phi(r_i \mid x_i, y_i, \theta) \cdot p_{\text{prior}}(\theta, \phi)$$

Here we are trying to model all the annotations, both y_i and r_i. The first factor predicts y_i using an ordinary probabilistic classifier p_θ, while the novel second factor predicts r_i using a model p_ϕ of how annotators generate the rationale annotations.

The crucial point is that the second factor depends on θ (since r_i is supposed to reflect the relation between x_i and y_i that is modeled by θ). As a result, the learner has an incentive to modify θ in a way that increases the second factor, even if this somewhat decreases the first factor on training data.[4]

After training, one should simply use the first factor $p_\theta(y \mid x)$ to classify test documents x. The second factor is irrelevant for test documents, since they have not been annotated with rationales r.

The second factor may likewise be omitted for any training documents i that have not been annotated with rationales, as there is no r_i to predict in those cases. In the extreme case where no documents are annotated with rationales, equation (1) reduces to the standard training procedure.

3.2 Noisy channel design of rationale models

Like ordinary class annotations, rationale annotations present us with a "credit assignment problem," albeit a smaller one that is limited to features that fire "in the vicinity" of the rationale r. Some of these θ-features were likely responsible for the classification y and hence triggered the rationale. Other such θ-features were just innocent bystanders.

Thus, the interesting part of our model is $p_\phi(r \mid x, y, \theta)$, which models the rationale annotation process. *The rationales r reflect θ, but in noisy ways.*

Taking this noisy channel idea seriously, $p_\phi(r \mid x, y, \theta)$ should consider two questions when assessing whether r is a plausible set of rationales given θ. First, it needs a "language model" of rationales: does r consist of rationales that are well-formed *a priori*, i.e., before θ is considered? Second, it needs a "channel model": does r faithfully signal the features of θ that strongly support classifying x as y?

If a feature contributes heavily to the classification of document x as class y, then the **channel model** should tell us which *parts* of document x tend to be highlighted as a result.

The channel model must know about the particular kinds of features that are extracted by f and scored by θ. Suppose the feature not . . . gripping,[5] with weight θ_h, is predictive of the annotated class y. This raises the probabilities of the annotator's highlighting each of various words, or combinations of words, in a phrase like not the most gripping banquet on film. The channel model parameters in ϕ

[3]It would be preferable to integrate out ϕ (and even θ), but more difficult.

[4]Interestingly, even examples where the annotation y_i is wrong or unhelpful can provide useful information about θ via the pair (y_i, r_i). Two annotators marking the same movie review might disagree on whether it is overall a positive or negative review—but the second factor still allows learning positive features from the first annotator's positive rationales, and negative features from the second annotator's negative rationales.

[5]Our current experiments use only unigram features, to match past work, but we use this example to outline how our approach generalizes to complex linguistic (or visual) features.

should specify how *much* each of these probabilities is raised, based on the magnitude of $\theta_h \in \mathbb{R}$, the class y, and the fact that the feature is an instance of the template <Neg> ... <Adjective>. (Thus, ϕ has no parameters specific to the word gripping; it is a low-dimensional vector that only describes the annotator's general style in translating θ into r.)

The **language model**, however, is independent of the feature set θ. It models what rationales tend to look like in the input domain—e.g., documents or images. In the document case, ϕ should describe: How frequent and how long are typical rationales? Do their edges tend to align with punctuation or major syntactic boundaries in x? Are they rarer in the middle of a document, or in certain documents?[6]

Thanks to the language model, we do not need to posit high θ features to explain every word in a rationale. The language model can "explain away" some words as having been highlighted only because this annotator prefers not to end a rationale in mid-phrase, or prefers to sweep up close-together features with a single long rationale rather than many short ones. Similarly, the language model can help explain why some words, though important, might *not* have been included in any rationale of r.

If there are multiple annotators, one can learn different ϕ parameters for each annotator, reflecting their different annotation styles.[7] We found this to be useful (section 8.2).

We remark that our generative modeling approach (equation (1)) would also apply if r were not rationale markup, but some other kind of so-called "side information," such as the *feature* annotations discussed in section 1. For example, Raghavan et al. (2006) assume that if feature h is relevant—a binary distinction—iff it was selected in at least one document. But it might be more informative to observe that h was selected in 3 of the 10 documents where it appeared, and to predict this via a model $p_\phi(3 \text{ of } 10 \mid \theta_h)$, where ϕ describes (e.g.) how to derive a binomial parameter nonlinearly from θ_h. This approach would not *how often* h was marked and infer *how relevant* is feature h (i.e., infer θ_h). In this case, p_ϕ is a simple channel that transforms relevant features into direct indicators of the feature. Our side information merely requires a more complex transformation—from relevant features into well-formed rationales, modulated by documents.

4 Experimental Data: Movie Reviews

In Zaidan et al. (2007), we introduced the "Movie Review Polarity Dataset Enriched with Annotator Rationales."[8] It is based on the dataset of Pang and Lee (2004),[9] which consists of 1000 positive and 1000 negative movie reviews, tokenized and divided into 10 folds (F_0–F_9). All our experiments use F_9 as their final blind test set.

The enriched dataset adds rationale annotations produced by an annotator A0, who annotated folds F_0–F_8 of the movie review set with rationales (in the form of textual substrings) that supported the gold-standard classifications. We will use A0's data to determine the improvement of our method over a (log-linear) baseline model without rationales. We also use A0 to compare against the "masking SVM" method and SVM baseline of Zaidan et al. (2007).

Since ϕ can be tuned to a particular annotator, we would also like to know how well this works with data from annotators other than A0. We randomly selected 100 reviews (50 positive and 50 negative) and collected both class and rationale annotation data from each of six new annotators A3–A8,[10] following the same procedures as (Zaidan et al., 2007). We report results using only data from A3–A5, since we used the data from A6–A8 as development data in the early stages of our work.

We use this new rationale-enriched dataset[8] to determine if our method works well across annotators. We will only be able to carry out that comparison

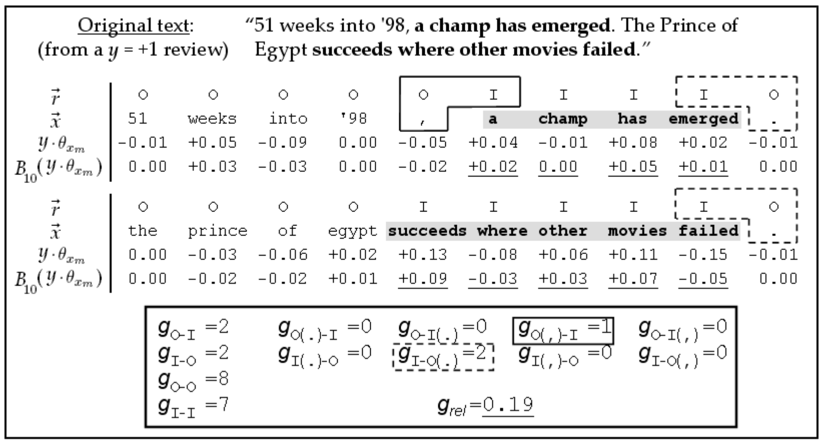
[6]Our current experiments do not model this last point. However, we imagine that if the document only has a few θ-features that support the classification, the annotator will probably mark most of them, whereas if such features are abundant, the annotator may lazily mark only a few of the strongest ones. A simple approach would equip ϕ with a different "bias" or "threshold" parameter ϕ_x for each rationale training document x, to modulate the *a priori* probability of marking a rationale in x. By fitting this bias parameter, we deduce how lazy the annotator was (for whatever reason) on document x. If desired, a prior on ϕ_x could consider whether x has many strong θ-features, whether the annotator has recently had a coffee break, etc.

[7]Given insufficient rationale data to recover some annotator's ϕ well, one could smooth using data from other annotators. But in our situation, ϕ had relatively few parameters to learn.

[8]Available at http://cs.jhu.edu/∼ozaidan/rationales.

[9]Polarity dataset version 2.0.

[10]We avoid annotator names A1–A2, which were already used in (Zaidan et al., 2007).



Original text: "51 weeks into '98, a champ has emerged. The Prince of Egypt succeeds where other movies failed."
(from a y = +1 review)
r⃗ | O O O O O I I I I O
x⃗ | 51 weeks into '98 , a champ has emerged .
y·θ_xm | -0.01 +0.05 -0.09 0.00 -0.05 +0.04 -0.01 +0.08 +0.02 -0.01
B_10(y·θ_xm) | 0.00 +0.03 -0.03 0.00 -0.02 +0.02 0.00 +0.05 +0.01 0.00
r⃗ | O O O O I I I I I O
x⃗ | the prince of egypt succeeds where other movies failed .
y·θ_xm | 0.00 -0.03 -0.06 +0.02 +0.13 -0.08 +0.06 +0.11 -0.15 -0.01
B_10(y·θ_xm) | 0.00 -0.02 -0.02 +0.01 +0.09 -0.03 +0.03 +0.07 -0.05 0.00
g_O-I =2
g_I-O =2
g_O-O =8
g_I-I =7
g_O(.)-I =0
g_I(.)-O =0
g_O-I(.) =0
g_I-O(.) =2
g_O(,)-I =1
g_I(,)-O =0
g_O-I(,) =0
g_I-O(,) =0
g_rel =0.19

Figure 1: Rationales as sequence annotation: the annotator highlighted two textual segments as rationales for a positive class. Highlighted words in $\vec{x}$ are tagged I in $\vec{r}$, and other words are tagged O. The figure also shows some ϕ-features. For instance, $g_{O(,)\text{-}I}$ is a count of O−I transitions that occur with a comma as the left word. Notice also that g_{rel} is the sum of the underlined values.

at small training set sizes, due to limited data from A3–A8. The larger A0 dataset will still allow us to evaluate our method on a range of training set sizes.

5 Detailed Models

5.1 Modeling class annotations with p_θ

We define the basic classifier p_θ in equation (1) to be a standard conditional log-linear model:

$$p_\theta(y \mid x) \overset{\text{def}}{=} \frac{\exp(\vec{\theta} \cdot \vec{f}(x, y))}{Z_\theta(x)} \overset{\text{def}}{=} \frac{u(x, y)}{Z_\theta(x)} \qquad (2)$$

where $\vec{f}(\cdot)$ extracts a feature vector from a classified document, $\vec{\theta}$ are the corresponding weights of those features, and $Z_\theta(x) \overset{\text{def}}{=} \sum_y u(x, y)$ is a normalizer.

We use the same set of binary features as in previous work on this dataset (Pang et al., 2002; Pang and Lee, 2004; Zaidan et al., 2007). Specifically, let $V = \{v_1, ..., v_{17744}\}$ be the set of word types with count ≥ 4 in the full 2000-document corpus. Define $f_h(x, y)$ to be y if v_h appears at least once in x, and 0 otherwise. Thus $\theta \in \mathbb{R}^{17744}$, and positive weights in θ favor class label $y = +1$ and equally discourage $y = -1$, while negative weights do the opposite.

This standard unigram feature set is linguistically impoverished, but serves as a good starting point for studying rationales. Future work should consider more complex features and how *they* are signaled by rationales, as discussed in section 3.2.

5.2 Modeling rationale annotations with p_ϕ

The rationales collected in this task are textual segments of a document to be classified. The document itself is a word token sequence $\vec{x} = x_1, ..., x_M$. We encode its rationales as a corresponding tag sequence $\vec{r} = r_1, ..., r_M$, as illustrated in Figure 1. Here $r_m \in \{\texttt{I}, \texttt{O}\}$ according to whether the token x_m is **in** a rationale (i.e., x_m was at least partly highlighted) or **outside** all rationales. x_1 and x_M are special boundary symbols, tagged with O.

We predict the full tag sequence $\vec{r}$ at once using a conditional random field (Lafferty et al., 2001). A CRF is just another conditional log-linear model:

$$p_\phi(r \mid x, y, \vec{\theta}) \overset{\text{def}}{=} \frac{\exp(\vec{\phi} \cdot \vec{g}(r, x, y, \vec{\theta}))}{Z_\phi(x, y, \vec{\theta})} \overset{\text{def}}{=} \frac{u(r, x, y, \vec{\theta})}{Z_\phi(x, y, \vec{\theta})}$$

where $\vec{g}(\cdot)$ extracts a feature vector, $\vec{\phi}$ are the corresponding weights of those features, and $Z_\phi(x, y, \vec{\theta}) \overset{\text{def}}{=} \sum_r u(r, x, y, \vec{\theta})$ is a normalizer.

As usual for linear-chain CRFs, $\vec{g}(\cdot)$ extracts two kinds of features: first-order "emission" features that relate r_m to (x_m, y, θ), and second-order "transition" features that relate r_m to r_{m-1} (although some of these also look at x).

These two kinds of features respectively capture the "channel model" and "language model" of section 3.2. The former says r_m is I because x_m is associated with a relevant θ-feature. The latter says r_m is I simply because it is next to another I.

5.3 Emission ϕ-features ("channel model")

Recall that our θ-features (at present) correspond to unigrams. Given $(\vec{x}, y, \vec{\theta})$, let us say that a unigram $w \in \vec{x}$ is **relevant**, **irrelevant**, or **anti-relevant** if $y \cdot \theta_w$ is respectively $\gg 0$, ≈ 0, or $\ll 0$. That is, w is relevant if its presence in x strongly supports the annotated class y, and anti-relevant if its presence strongly supports the opposite class $-y$.

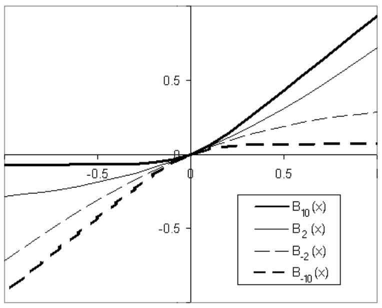


Figure 2: The function family B_s in equation (3), shown for $s \in \{10, 2, -2, -10\}$.

We would like to learn the extent ϕ_{rel} to which annotators try to *include* relevant unigrams in their rationales, and the (usually lesser) extent ϕ_{antirel} to which they try to *exclude* anti-relevant unigrams. This will help us infer $\vec{\theta}$ from the rationales.

The details are as follows. ϕ_{rel} and ϕ_{antirel} are the weights of two emission features extracted by $\vec{g}$:

$$g_{\text{rel}}(\vec{x}, y, \vec{r}, \vec{\theta}) \stackrel{\text{def}}{=} \sum_{m=1}^{M} I(r_m = \mathtt{I}) \cdot B_{10}(y \cdot \theta_{x_m})$$

$$g_{\text{antirel}}(\vec{x}, y, \vec{r}, \vec{\theta}) \stackrel{\text{def}}{=} \sum_{m=1}^{M} I(r_m = \mathtt{I}) \cdot B_{-10}(y \cdot \theta_{x_m})$$

Here $I(\cdot)$ denotes the indicator function, returning 1 or 0 according to whether its argument is true or false. Relevance and negated anti-relevance are respectively measured by the differentiable nonlinear functions B_{10} and B_{-10}, which are defined by

$$B_s(a) = (\log(1 + \exp(a \cdot s)) - \log(2))/s \qquad (3)$$

and graphed in Figure 2. Sample values of B_{10} and g_{rel} are shown in Figure 1.

How does this work? The g_{rel} feature is a sum over all unigrams in the document $\vec{x}$. It does not fire strongly on the irrelevant or anti-relevant unigrams, since B_{10} is close to zero there.[11] But it fires positively on relevant unigrams w if they are tagged with I, and the strength of such firing increases approximately linearly with θ_w. Since the weight $\phi_{\text{rel}} > 0$ in practice, this means that raising a relevant unigram's θ_w (if $y = +1$) will proportionately raise its log-odds of being tagged with I. Symmetrically, since $\phi_{\text{antirel}} > 0$ in practice, lowering an anti-relevant unigram's θ_w (if $y = +1$) will proportionately lower its log-odds of being tagged with I, though not necessarily at the same rate as for relevant unigrams.[12]

Should ϕ also include traditional CRF emission features, which would recognize that particular words like great tend to be tagged as I? No! Such features would undoubtedly do a better job predicting the rationales and hence increasing equation (1). However, crucially, our true goal is not to predict the rationales but to recover the classifier parameters θ. Thus, if great tends to be highlighted, then the model should not be permitted to explain this directly by increasing some feature ϕ_{great}, but only indirectly by increasing θ_{great}. We therefore permit our rationale prediction model to consider only the two emission features g_{rel} and g_{antirel}, which see the words in $\vec{x}$ only through their θ-values.

5.4 Transition ϕ-features ("language model")

Annotators highlight more than just the relevant unigrams. (After all, they aren't told that our current θ-features are unigrams.) They tend to mark full phrases, though perhaps taking care to exclude anti-relevant portions. ϕ models these phrases' shape, via weights for several "language model" features.

Most important are the 4 traditional CRF tag transition features $g_{\mathtt{O}\text{-}\mathtt{O}}, g_{\mathtt{O}\text{-}\mathtt{I}}, g_{\mathtt{I}\text{-}\mathtt{I}}, g_{\mathtt{I}\text{-}\mathtt{O}}$. For example, $g_{\mathtt{O}\text{-}\mathtt{I}}$ *counts* the number of O-to-I transitions in $\vec{r}$ (see Figure 1). Other things equal, an annotator with high $\phi_{\mathtt{O}\text{-}\mathtt{I}}$ is predicted to have many rationales per 1000 words. And if $\phi_{\mathtt{I}\text{-}\mathtt{I}}$ is high, rationales are predicted to be long phrases (including more irrelevant unigrams around or between the relevant ones).

We also learn more refined versions of these features, which consider how the transition probabilities are influenced by the punctuation and syntax of the document $\vec{x}$ (independent of $\vec{\theta}$). These refined features are more specific and hence more sparsely trained. Their weights reflect deviations from the simpler, "backed-off" transition features such as $g_{\mathtt{O}\text{-}\mathtt{I}}$. (Again, see Figure 1 for examples.)

Conditioning on left word. A feature of the form $g_{t_1(v)\text{-}t_2}$ is specified by a pair of tag types $t_1, t_2 \in \{\mathtt{I}, \mathtt{O}\}$ and a vocabulary word type v. It counts the

[11] B_{10} sets the threshold for relevance to be about 0. One could also include versions of the g_{rel} feature that set a higher threshold, using $B_{10}(y \cdot \theta_{x_m} - \text{threshold})$.

[12] If the two rates *are* equal ($\phi_{\text{rel}} = \phi_{\text{antirel}}$), we get a simpler model in which the log-odds change exactly linearly with θ_w for each w, regardless of w's relevance/irrelevance/anti-relevance. This follows from the fact that $B_s(a) + B_{-s}(a)$ simplifies to a.

number of times an t_1–t_2 transition occurs in $\vec{r}$ conditioned on v appearing as the first of the two word tokens where the transition occurs. Our experiments include $g_{t_1(v)\text{-}t_2}$ features that tie `I-O` and `O-I` transitions to the 4 most frequent punctuation marks v (comma, period, ?, !).

Conditioning on right word. A feature $g_{t_1\text{-}t_2(v)}$ is similar, but v must appear as the second of the two word tokens where the transition occurs. Again here, we use $g_{t_1\text{-}t_2(v)}$ features that tie `I-O` and `O-I` transitions to the four punctuation marks mentioned above. We also include five features that tie `O-I` transitions to the words *no*, *not*, *so*, *very*, and *quite*, since in our development data, those words were more likely than others to start rationales.[13]

Conditioning on syntactic boundary. We parsed each rationale-annotated training document (no parsing is needed at test time).[14] We then marked each word bigram x_1-x_2 with three nonterminals: N_{End} is the nonterminal of the largest constituent that contains x_1 and not x_2, N_{Start} is the nonterminal of the largest constituent that contains x_2 and not x_1, and N_{Cross} is the nonterminal of the *smallest* constituent that contains both x_1 and x_2.

For a nonterminal N and pair of tag types (t_1, t_2), we define three features, $g_{t_1\text{-}t_2/\text{E}=N}$, $g_{t_1\text{-}t_2/\text{S}=N}$, and $g_{t_1\text{-}t_2/\text{C}=N}$, which count the number of times a t_1-t_2 transition occurs in $\vec{r}$ with N matching the N_{End}, N_{Start}, or N_{Cross} nonterminal, respectively. Our experiments include these features for 11 common nonterminal types N (DOC, TOP, S, SBAR, FRAG, PRN, NP, VP, PP, ADJP, QP).

6 Training: Joint Optimization of θ and ϕ

To train our model, we use L-BFGS to locally maximize the log of the objective function (1):[15]

$$\sum_{i=1}^{n} \log p_\theta(y_i \mid x_i) - \frac{1}{2\sigma_\theta^2}\|\theta\|^2$$
$$+C\Big(\sum_{i=1}^{n} \log p_\phi(r_i \mid x_i, y_i, \theta)\Big) - \frac{1}{2\sigma_\phi^2}\|\phi\|^2 \qquad (4)$$

This defines p_{prior} from (1) to be a standard diagonal Gaussian prior, with variances σ_θ^2 and σ_ϕ^2 for the two sets of parameters. We optimize σ_θ^2 in our experiments. As for σ_ϕ^2, different values did not affect the results, since we have a large number of $\{$`I`,`O`$\}$ rationale tags to train relatively few ϕ weights; so we simply use $\sigma_\phi^2 = 1$ in all of our experiments.

Note the new C factor in equation (4). Our initial experiments showed that optimizing equation (4) without C led to an increase in the likelihood of the rationale data at the expense of classification accuracy, which degraded noticeably. This is because the second sum in (4) has a much larger magnitude than the first: in a set of 100 documents, it predicts around 74,000 binary $\{$`I`,`O`$\}$ tags, versus the one hundred binary class labels. While we are willing to reduce the log-likelihood of the training classifications (the first sum) to a certain extent, focusing too much on modeling rationales (the second sum) is clearly not our ultimate goal, and so we optimize C on development data to achieve some balance between the two terms of equation (4). Typical values of C range from $\frac{1}{300}$ to $\frac{1}{50}$.[16]

We perform alternating optimization on θ and ϕ:

1. Initialize θ to maximize equation (4) but with $C = 0$ (i.e. based only on class data).
2. Fix θ, and find ϕ that maximizes equation (4).
3. Fix ϕ, and find θ that maximizes equation (4).
4. Repeat 2 and 3 until convergence.

The L-BFGS method requires calculating the gradient of the objective function (4). The partial derivatives with respect to components of θ and ϕ involve calculating expectations of the feature functions, which can be computed in linear time (with respect to the size of the training set) using the forward-backward algorithm for CRFs. The partial derivatives also involve the derivative of (3), to determine how changing θ will affect the firing strength of the emission features g_{rel} and g_{antirel}.

[13]These are the function words with count ≥ 40 in a random sample of 100 documents, and which were associated with the `O-I` tag transition at more than twice the average rate. We do not use any other lexical ϕ-features that reference $\vec{x}$, for fear that they would enable the learner to explain the rationales without changing θ as desired (see the end of section 5.3).

[14]We parse each sentence with the Collins parser (Collins, 1999). Then the document has one big parse tree, whose root is `DOC`, with each sentence being a child of `DOC`.

[15]One might expect this function to be convex because p_θ and p_ϕ are both log-linear models with no hidden variables. However, $\log p_\phi(r_i \mid x_i, y_i, \theta)$ is not necessarily convex in θ.

[16]C also balances our confidence in the classifications y against our confidence in the rationales r; either may be noisy.

7 Experimental Procedures

We report on two sets of experiments. In the first set, we use the annotation data that A3–A5 provided for the small set of 100 documents (as well as the data from A0 on those same 100 documents). In the second set, we used A0's abundant annotation data to evaluate our method with training set sizes up to 1600 documents, and compare it with three other methods: log-linear baseline, SVM baseline, and the SVM masking method of (Zaidan et al., 2007).

7.1 Learning curves

The learning curves reported in section 8.1 are generated exactly as in (Zaidan et al., 2007). Each curve shows classification accuracy at training set sizes $T = 1, 2, ..., 9$ folds (i.e. $200, 400, ..., 1600$ training documents). For a given size T, the reported accuracy is an average of 9 experiments with different subsets of the entire training set, each of size T:

$$\frac{1}{9}\sum_{i=0}^{8} acc(F_9 \mid F_{i+1} \cup \ldots \cup F_{i+T}) \qquad (5)$$

where F_j denotes the fold numbered $j \bmod 9$, and $acc(F_9 \mid Y)$ means classification accuracy on the held-out test set F_9 after training on set Y.

We use an appropriate paired permutation test, detailed in (Zaidan et al., 2007), to test differences in (5). We call a difference significant at $p < 0.05$.

7.2 Comparison to "masking SVM" method

We compare our method to the "masking SVM" method of (Zaidan et al., 2007). Briefly, that method used rationales to construct several so-called *contrast examples* from every training example. A contrast example is obtained by "masking out" one of the rationales highlighted to support the training example's class. A *good* classifier should have more trouble on this modified example. Hence, Zaidan et al. (2007) required the learned SVM to classify each contrast example with a smaller margin than the corresponding original example (and did not require it to be classified correctly).

The masking SVM learner relies on a simple geometric principle; is trivial to implement on top of an existing SVM learner; and works well. However, we believe that the generative method we present here is more interesting and should apply more broadly.

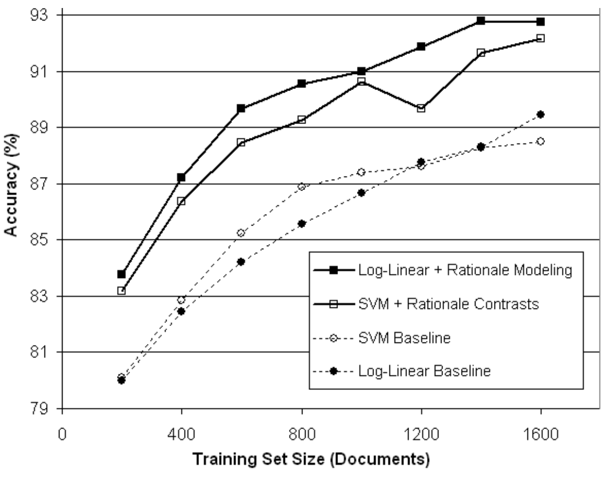


Figure 3: Classification accuracy curves for the 4 methods: the two baseline learners that only utilize class data, and the two learners that also utilize rationale annotations. The SVM curves are from (Zaidan et al., 2007).

First, the masking method is specific to improving an SVM learner, whereas our method can be used to improve any classifier by adding a rationale-based regularizer (the second half of equation (4)) to its objective function during training.

More important, there are tasks where it is unclear how to generate contrast examples. For the movie review task, it was natural to mask out a rationale by pretending its words never occurred in the document. After all, most word types do not appear in most documents, so it is natural to consider the non-presence of a word as a "default" state to which we can revert. But in an image classification task, how should one modify the image's features to ignore some spatial region marked as a rationale? There is usually no natural "default" value to which we could set the pixels. Our method, on the other hand, eliminates contrast examples altogether.

8 Experimental Results and Analysis

8.1 The added benefit of rationales

Fig. 3 shows learning curves for four methods. A log-linear model shows large and significant improvements, at all training sizes, when we incorporate rationales into its training via equation (4). Moreover, the resulting classifier consistently outperforms[17] prior work, the masking SVM, which starts with a slightly better baseline classifier (an SVM) but incorporates the rationales more crudely.

[17]Differences are not significant at sizes 200, 1000, and 1600.

	size	A0	A3	A4	A5
SVM baseline	100	72.0	72.0	72.0	70.0
SVM+contrasts	100	*75.0*	*73.0*	*74.0*	*72.0*
Log-linear baseline	100	71.0	*73.0*	71.0	70.0
Log-linear+rats	100	**76.0**	**76.0**	**77.0**	**74.0**
SVM baseline	20	63.4	*62.2*	60.4	62.6
SVM+contrasts	20	*65.4*	*63.4*	*62.4*	**64.8**
Log-linear baseline	20	63.0	*62.2*	60.2	62.4
Log-linear+rats	20	**65.8**	**63.6**	**63.4**	**64.8**

Table 1: Accuracy rates using each annotator's data. In a given column, a value in *italics* is not significantly different from the highest value in that column, which is **boldfaced**. The size=20 results average over 5 experiments.

To confirm that we could successfully model annotators other than A0, we performed the same comparison for annotators A3–A5; each had provided class and rationale annotations on a small 100-document training set. We trained a separate ϕ for each annotator. Table 1 shows improvements over baseline, usually significant, at 2 training set sizes.

8.2 Analysis

Examining the learned weights $\vec{\phi}$ gives insight into annotator behavior. High weights include `I-O` and `O-I` transitions conditioned on punctuation, e.g., $\phi_{\texttt{I(.)-O}} = 3.55$,[18] as well as rationales ending at the end of a major phrase, e.g., $\phi_{\texttt{I-O/E=VP}} = 1.88$.

The large emission feature weights, e.g., $\phi_{\text{rel}} = 14.68$ and $\phi_{\text{antirel}} = 15.30$, tie rationales closely to θ values, as hoped. For example, in Figure 1, the word w = succeeds, with $\theta_w = 0.13$, drives up $p(\texttt{I})/p(\texttt{O})$ by a factor of 7 (in a positive document) relative to a word with $\theta_w = 0$.

In fact, feature ablation experiments showed that almost all the classification benefit from rationales can be obtained by using only these 2 emission ϕ-features and the 4 unconditioned transition ϕ-features. Our full ϕ (115 features) merely improves our ability to predict the rationales (whose likelihood does increase significantly with more features).

We also checked that annotators' styles differ enough that it helps to tune ϕ to the "target" annotator A who gave the rationales. Table 3 shows that a ϕ model trained on A's *own* rationales does best at predicting new rationales from A. Table 2 shows that as

[18]When trained on folds F_4–F_8 with A0's rationales.

	ϕ_{A0}	ϕ_{A3}	ϕ_{A4}	ϕ_{A5}	Baseline
θ_{A0}	**76.0**	*73.0*	*74.0*	*73.0*	71.0
θ_{A3}	*73.0*	**76.0**	*74.0*	*73.0*	*73.0*
θ_{A4}	*75.0*	73.0	**77.0**	*74.0*	71.0
θ_{A5}	**74.0**	*71.0*	*72.0*	**74.0**	70.0

Table 2: Accuracy rate for an annotator's θ (rows) obtained when using some other annotator's ϕ (columns). Notice that the diagonal entries and the baseline column are taken from rows of Table 1 (size=100).

	ϕ_{A0}	ϕ_{A3}	ϕ_{A4}	ϕ_{A5}	Trivial model
$-L(r_{A0})$	**0.073**	0.086	*0.077*	0.088	0.135
$-L(r_{A3})$	0.084	**0.068**	*0.071*	**0.068**	0.130
$-L(r_{A4})$	0.088	0.084	**0.075**	0.085	0.153
$-L(r_{A5})$	0.058	**0.044**	*0.047*	**0.044**	0.111

Table 3: Cross-entropy per tag of rationale annotations $\vec{r}$ for each annotator (rows), when predicted from that annotator's $\vec{x}$ and $\vec{\theta}$ via a possibly different annotator's ϕ (columns). For comparison, the trivial model is a bigram model of $\vec{r}$, which is trained on the target annotator but ignores $\vec{x}$ and $\vec{\theta}$. 5-fold cross-validation on the 100-document set was used to prevent testing on training data.

a result, classification performance on the test set is usually best if it was A's *own* ϕ that was used to help learn θ from A's rationales. In both cases, however, a different annotator's ϕ is better than nothing.

9 Conclusions

We have demonstrated a effective method for eliciting extra knowledge from *naive* annotators, in the form of lightweight "rationales" for their annotations. By explicitly modeling the annotator's rationale-marking process, we are able to infer a better model of the original annotations.

We showed that our method performs significantly better than two strong baseline classifiers, and also outperforms our previous discriminative method for exploiting rationales (Zaidan et al., 2007). We also saw that it worked across four annotators who have different rationale-marking styles.

In future, we are interested in new domains that can adaptively solicit rationales for some or all training examples. Our new method, being essentially Bayesian inference, is potentially extensible to many other situations—other tasks, classifier architectures, and more complex features.

References

Eric Brill and Grace Ngai. 1999. Man [and woman] vs. machine: A case study in base noun phrase learning. In *Proceedings of the 37th ACL Conference*.

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

G. Druck, G. Mann, and A. McCallum. 2008. Learning from labeled features using generalized expectation criteria. In *Proceedings of ACM Special Interest Group on Information Retrieval, (SIGIR)*.

A. Haghighi and D. Klein. 2006. Prototype-driven learning for sequence models. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 320–327, New York City, USA, June. Association for Computational Linguistics.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*.

Grace Ngai and David Yarowsky. 2000. Rule writing or annotation: Cost-efficient resource usage for base noun phrase chunking. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 117–125, Hong Kong.

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.

Hema Raghavan and James Allan. 2007. An interactive algorithm for asking and incorporating feature feedback into support vector machines. In *Proceedings of SIGIR*.

Hema Raghavan, Omid Madani, and Rosie Jones. 2006. Active learning on both features and instances. *Journal of Machine Learning Research*, 7:1655–1686, Aug.

Luis von Ahn, Ruoran Liu, and Manuel Blum. 2006. Peekaboom: A game for locating objects. In *CHI '06: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 55–64.

Omar Zaidan, Jason Eisner, and Christine Piatko. 2007. Using "annotator rationales" to improve machine learning for text categorization. In *NAACL HLT 2007; Proceedings of the Main Conference*, pages 260–267, April.

One-Class Clustering in the Text Domain

Ron Bekkerman
HP Laboratories
Palo Alto, CA 94304, USA
ron.bekkerman@hp.com

Koby Crammer
University of Pennsylvania
Philadelphia, PA 19104, USA
crammer@cis.upenn.edu

Abstract

Having seen a news title "Alba denies wedding reports", how do we infer that it is primarily about Jessica Alba, rather than about weddings or reports? We probably realize that, in a randomly driven sentence, the word "Alba" is less anticipated than "wedding" or "reports", which adds value to the word "Alba" if used. Such anticipation can be modeled as a ratio between an empirical probability of the word (in a given corpus) and its estimated probability in general English. Aggregated over all words in a document, this ratio may be used as a measure of the document's topicality. Assuming that the corpus consists of on-topic and off-topic documents (we call them *the core* and *the noise*), our goal is to determine which documents belong to the core. We propose two unsupervised methods for doing this. First, we assume that words are sampled i.i.d., and propose an information-theoretic framework for determining the core. Second, we relax the independence assumption and use a simple graphical model to rank documents according to their likelihood of belonging to the core. We discuss theoretical guarantees of the proposed methods and show their usefulness for Web Mining and Topic Detection and Tracking (TDT).

1 Introduction

Many intelligent applications in the text domain aim at determining whether a document (a sentence, a snippet etc.) is on-topic or off-topic. In some applications, topics are explicitly given. In binary text classification, for example, the topic is described in terms of positively and negatively labeled documents. In information retrieval, the topic is imposed by a query. In many other applications, the topic is unspecified, however, its existence is assumed. Examples of such applications are within text summarization (extract the most topical sentences), text clustering (group documents that are close topically), novelty detection (reason whether or not test documents are on the same topic as training documents), spam filtering (reject incoming email messages that are too far topically from the content of a personal email repository), etc.

Under the (standard) Bag-Of-Words (BOW) representation of a document, *words* are the functional units that bear the document's topic. Since some words are topical and some are not, the problem of detecting on-topic documents has a dual formulation of detecting topical words. This paper deals with the following questions: (a) Which words can be considered topical? (b) How can topical words be detected? (c) How can on-topic documents be detected given a set of topical words?

The BOW formalism is usually translated into the generative modeling terms by representing documents as multinomial word distributions. For the on-topic/off-topic case, we assume that words in a document are sampled from a mixture of two multinomials: one over topical words and another one over general English (i.e. the background). Obviously enough, the support of the "topic" multinomial is significantly smaller than the support of the background. A document's topicality is then determined by aggregating the topicality of its words (see below for details). Note that by introducing the background distribution we refrain from explicitly modeling the class of off-topic documents—a document is supposed to be off-topic if it is "not topical enough".

Such a formulation of topicality prescribes using the *one-class* modeling paradigm, as opposed to sticking to the *binary* case. Besides being much

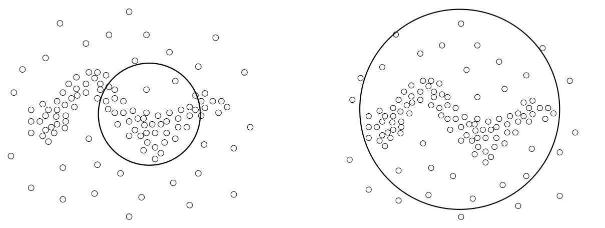


Figure 1: The problem of hyperspherical decision boundaries in one-class models for text, as projected on 2D: (left) a too small portion of the core is captured; (right) too much space around the core is captured.

less widely studied and therefore much more attractive from the scientific point of view, one-class models appear to be more adequate for many real-world tasks, where negative examples are not straightforwardly observable. One-class models separate the desired class of data instances (*the core*) from other data instances (*the noise*). Structure of noise is either unknown, or too complex to be explicitly modeled.

One-class problems are traditionally approached using vector-space methods, where a convex decision boundary is built around the data instances of the desired class, separating it from the rest of the universe. In the text domain, however, those vector-space models are questionably applicable—unlike effective *binary* vector-space models. In binary models, decision boundaries are linear[1], whereas in (vector-space) one-class models, the boundaries are usually *hyperspherical*. Intuitively, since core documents tend to lie on a lower-dimensional manifold (Lebanon, 2005), inducing hyperspherical boundaries may be sub-optimal as they tend to either capture just a portion of the core, or capture too much space around it (see illustration in Figure 1). Here we propose alternative ways for detecting the core, which work well in text.

One-class learning problems have been studied as either outlier detection or identifying a small coherent subset. In one-class outlier detection (Tax and Duin, 2001; Schölkopf et al., 2001), the goal is to identify *a few* outliers from the given set of examples, where the vast majority of the examples are considered relevant. Alternatively, a complementary goal is to distill a subset of relevant examples, in the space with many outliers (Crammer and Chechik, 2004; Gupta and Ghosh, 2005; Crammer et al., 2008). Most of the one-class approaches employ geometrical concepts to capture the notion of relevancy (or irrelevancy) using either hyperplanes (Schölkopf et al., 2001) or hyperspheres (Tax and Duin, 2001; Crammer and Chechik, 2004; Gupta and Ghosh, 2005). In this paper we adopt the latter approach: we formulate one-class clustering in text as an optimization task of identifying the *most coherent* subset (*the core*) of k documents drawn from a given pool of $n > k$ documents.[2]

Given a collection $\mathcal{D}$ of on-topic and off-topic documents, we assume that on-topic documents share a portion of their vocabulary that consists of "relatively rare" words, i.e. words that are used in $\mathcal{D}$ more often than they are used in general English. We call them *topical* words. For example, if some documents in $\mathcal{D}$ share words such as "Bayesian", "classifier", "reinforcement" and other machine learning terms (infrequent in general English), whereas other documents do not seem to share any subset of words (besides stopwords), then we conclude that the machine learning documents compose the core of $\mathcal{D}$, while non-machine learning documents are noise.

We express the level of topicality of a word w in terms of the ratio $\rho(w) = \frac{p(w)}{q(w)}$, where $p(w)$ is w's empirical probability (in $\mathcal{D}$), and $q(w)$ is its estimated probability in general English. We discuss an interesting characteristic of $\rho(w)$: if $\mathcal{D}$ is large enough, then, with high probability, $\rho(w)$ values are greater for topical words than for non-topical words. Therefore, $\rho(w)$ can be used as a mean to measure the topicality of w.

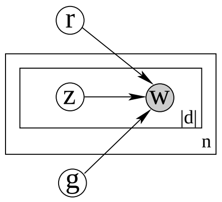
Obviously, the quality of this measure depends on the quality of estimating $q(w)$, i.e. the general English word distribution, which is usually estimated over a large text collection. The larger the collection is, the better would be the estimation. Recently, Google has released the `Web 1T` dataset[3] that provides $q(w)$ estimated on a text collection of one trillion tokens. We use it in our experimentation.

We propose two methods that use the ρ ratio to

[1]As such, or after applying the *kernel trick* (Cristianini and Shawe-Taylor, 2000)

[2]The parameter k is analogous to the number of clusters in (multi-class) clustering, as well as to the number of outliers (Tax and Duin, 2001) or the radius of Bregmanian ball (Crammer and Chechik, 2004)—in other formulations of one-class clustering.

[3]`http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2006T13`



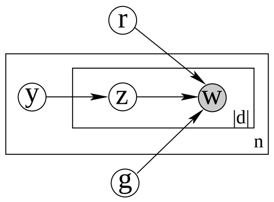


Figure 2: (left) A simple generative model; (right) Latent Topic/Background model (Section 4).

solve the one-class clustering problem. First, we express documents' topicality in terms of aggregating their words' ρ ratios into an information-theoretic "topicality measure". The core is then composed of k documents with the highest topicality measure. We show that the proposed measure is optimal for constructing the core cluster among documents of equal length. However, our method is not useful in a setup where some long documents have a topical *portion*: such documents should be considered on-topic, but their heavy tail of background words overcomes the topical words' influence. We generalize our method to non-equally-long documents by first extracting words that are supposed to be topical and then projecting documents over those words. Such projection preserves the optimality characteristic and results in constructing a more accurate core cluster in practice. We call such a method of choosing both topical words and core documents *One-Class Co-Clustering (OCCC)*.

It turns out that our OCCC method's performance depends heavily on choosing the number of topical words. We propose a heuristic for setting this number. As another alternative, we propose a method that does not require tuning this parameter: we use words' ρ ratios to initialize an EM algorithm that computes the likelihood of documents to belong to the core—we then choose k documents of maximal likelihood. We call this model the *Latent Topic/Background (LTB)* model. LTB outperforms OCCC in most of our test cases.

Our one-class clustering models have interesting cross-links with models applied to other Information Retrieval tasks. For example, a model that resembles our OCCC, is proposed by Zhou and Croft (2007) for *query performance prediction*. Tao and Zhai (2004) describe a *pseudo-relevance feedback* model that is similar to our LTB. These types of cross-links are common for the models that are

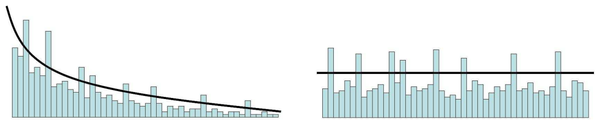


Figure 3: (left) Words' $p(w)$ values when sorted by their $q(w)$ values; (right) words' $\rho(w)$ values.

general enough and relatively simple. In this paper we put particular emphasis on the simplicity of our models, such that they are feasible for theoretical analysis as well as for efficient implementation.

2 Motivation for using ρ ratios

Recall that we use the $\rho(w) = \frac{p(w)}{q(w)}$ ratios to express the level of our "surprise" of seeing the word w. A high value of $\rho(w)$ means that w is used in the corpus more frequently than in general English, which, we assume, implies that w is topical. The more topical words a document contains, the more "topical" it is—k most topical documents compose the core $\mathcal{D}^k \subset \mathcal{D}$.

An important question is whether or not the ρ ratios are sufficient to detecting the *actually* topical words. To address this question, let us model the corpus $\mathcal{D}$ using a simple graphical model (Figure 2 left). In this model, the word distribution $p(w)$ is represented as a mixture of two multinomial distributions: p_r over a set $\mathcal{R}$ of topical words, and p_g over all the words $\mathcal{G} \supset \mathcal{R}$ in $\mathcal{D}$. For each word w_{ij} in a document d_i, we toss a coin Z_{ij}, such that, if $Z_{ij} = 1$, then w_{ij} is sampled from p_r, otherwise it is sampled from p_g. Define $\pi \triangleq p(Z_{ij} = 1)$.

If $|\mathcal{G}| \gg |\mathcal{R}| \gg 0$, and if $\pi \gg 0$, then topical words would tend to appear more often than non-topical words. However, we cannot simply base our conclusions on word counts, as some words are naturally more frequent than others (in general English). Figure 3 (left) illustrates this observation: it shows words' $p(w)$ values sorted by their $q(w)$ values. It is hard to fit a curve that would separate between $\mathcal{R}$ and $\mathcal{G} \setminus \mathcal{R}$. We notice however, that we can "flatten" this graph by drawing $\rho(w)$ values instead (see Figure 3 right). Here, naturally frequent words are penalized by the q factor, so we can assume that, when re-normalized, $\rho(w)$ behaves as a mixture of two discrete *uniform* distributions. A simple threshold can then separate between $\mathcal{R}$ and $\mathcal{G} \setminus \mathcal{R}$.

Proposition 1 *Under the uniformity assumption, it is sufficient to have a* log-linear *size sample (in $|\mathcal{G}|$) in order to determine the set $\mathcal{R}$ with high probability.*

See Bekkerman (2008) for the proof. The proposition states that in corpora of practical size[4] the set of topical words can be almost perfectly detected, simply by taking words with the highest ρ ratios. Consequently, the core $\mathcal{D}^k$ will consist of k documents, each of which contains more topical words than any document from $\mathcal{D} \setminus \mathcal{D}^k$.

To illustrate this theoretical result, we followed the generative process as described above, and constructed an artificial dataset with characteristics similar to those of our WAD dataset (see Section 5.1). In particular, we fixed the size of the artificial dataset to be equal to the size of the WAD dataset ($N = 330,000$). We set the ratio of topical words to 0.2 and assumed uniformity of the ρ values. In this setup, we were able to detect the set of topical words with a 98.5% accuracy.

2.1 Max-KL Algorithm

In this section, we propose a simple information-theoretic algorithm for identifying the core $\mathcal{D}^k$, and show that it is optimal under the uniformity assumption. Given the ρ ratios of words, the *aggregated topicality* of the corpus $\mathcal{D}$ can be expressed in terms of the KL-divergence:

$$\begin{aligned} KL(p||q) &= \sum_{w \in \mathcal{G}} p(w) \log \frac{p(w)}{q(w)} \\ &= \sum_{d \in \mathcal{D}, w \in \mathcal{G}} p(d,w) \log \frac{p(w)}{q(w)}. \end{aligned}$$

A document d's contribution to the aggregated topicality measure will assess the topicality of d:

$$KL_d(p||q) = \sum_{w \in \mathcal{G}} p(d,w) \log \frac{p(w)}{q(w)}. \qquad (1)$$

The core $\mathcal{D}^k$ will be composed of documents with the highest topicality scores. A simple, greedy algorithm for detecting $\mathcal{D}^k$ is then:

1. Sort documents according to their topicality value (1), in decreasing order.
2. Select the first k documents.

[4] $N = O(m \log m)$, where N is the number of word tokens in $\mathcal{D}$, and $m = |\mathcal{G}|$ is the size of the vocabulary.

Since the algorithm chooses documents with high values of the KL divergence we call it the *Max-KL* algorithm. We now argue that it is optimal under the uniformity assumption. Indeed, if the corpus $\mathcal{D}$ is large enough, then according to Proposition 1 (with high probability) any topical word w has a lower ρ ratio than any non-topical word. Assume that all documents are of the same length ($|d|$ is constant). The Max-KL algorithm chooses documents that contain more topical words than any other document in the corpus—which is exactly the definition of the core, as presented in Section 1. We summarize this observation in the following proposition:

Proposition 2 *If the corpus $\mathcal{D}$ is large enough, and all the documents are of the same length, then the Max-KL algorithm is optimal for the one-class clustering problem under the uniformity assumption.*

In contrast to the (quite natural) uniformity assumption, the all-the-same-length assumption is quite restrictive. Let us now propose an algorithm that overcomes this issue.

3 One-Class Co-Clustering (OCCC)

As accepted in Information Retrieval, we decide that a document is on-topic if it has a topical *portion*, no matter how long its non-topical portion is. Therefore, we decide about documents' topicality based on topical words only—non-topical words can be completely disregarded. This observation leads us to proposing a one-class *co-clustering* (OCCC) algorithm: we first detect the set $\mathcal{R}$ of topical words, represent documents over $\mathcal{R}$, and then detect $\mathcal{D}^k$ based on the new representation.[5]

We reexamine the document's topicality score (1) and omit non-topical words. The new score is then:

$$KL_d^r(p||q) = \sum_{w \in \mathcal{R}} p'(d,w) \log \frac{p(w)}{q(w)}, \qquad (2)$$

where $p'(d,w) = p(d,w)/(\sum_{w \in \mathcal{R}} p(d,w))$ is a joint distribution of documents and (only) topical words. The OCCC algorithm first uses $\rho(w)$ to

[5] OCCC is the simplest, *sequential* co-clustering algorithm, where words are clustered prior to clustering documents (see, e.g., Slonim and Tishby (2000)). In OCCC, word clustering is analogous to *feature selection*. More complex algorithms can be considered, where this analogy is less obvious.

choose the most topical words, then it projects documents on these words and apply the *Max-KL* algorithm, as summarized below:

1. Sort words according to their ρ ratios, in decreasing order.
2. Select a subset $\mathcal{R}$ of the first m_r words.
3. Represent documents as bags-of-words over $\mathcal{R}$ (delete counts of words from $\mathcal{G} \setminus \mathcal{R}$).
4. Sort documents according to their topicality score (2), in decreasing order.
5. Select a subset $\mathcal{D}^k$ of the first k documents.

Considerations analogous to those presented in Section 2.1, lead us to the following result:

Proposition 3 *If the corpus $\mathcal{D}$ is large enough, the OCCC algorithm is optimal for one-class clustering of documents, under the uniformity assumption.*

Despite its simplicity, the OCCC algorithm shows excellent results on real-world data (see Section 5). OCCC's time complexity is particularly appealing: $O(N)$, where N is the number of word tokens in $\mathcal{D}$.

3.1 Choosing size m_r of the word cluster

The choice of $m_r = |\mathcal{R}|$ can be crucial. We propose a useful heuristic for choosing it. We assume that the distribution of ρ ratios for $w \in \mathcal{R}$ is a Gaussian with a mean $\mu_r \gg 1$ and a variance σ_r^2, and that the distribution of ρ ratios for $w \in \mathcal{G} \setminus \mathcal{R}$ is a Gaussian with a mean $\mu_{nr} = 1$ and a variance σ_{nr}^2. We also assume that *all* the words with $\rho(w) < 1$ are non-topical. Since Gaussians are symmetric, we further assume that the number of non-topical words with $\rho(w) < 1$ equals the number of non-topical words with $\rho(w) \geq 1$. Thus, our estimate of $|\mathcal{G} \setminus \mathcal{R}|$ is twice the number of words with $\rho(w) < 1$, and then the number of topical words can be estimated as $m_r = |\mathcal{G}| - 2 \cdot \#\{\text{words with } \rho(w) < 1\}$.

4 Latent Topic/Background (LTB) model

Instead of sharply thresholding topical and non-topical words, we can have them all, *weighted* with a probability of being topical. Also, we notice that our original generative model (Figure 2 left) assumes that words are i.i.d. sampled, which can be relaxed by deciding on the document topicality first. In our new generative model (Figure 2 right), for each document d_i, Y_i is a Bernoulli random variable where

Algorithm 1 EM algorithm for one-class clustering using the LTB model.

```
Input:
  $\mathcal{D}$ – the dataset
  $\rho(w_l) = \frac{p(w_l)}{q(w_l)}$ – $\rho$ scores for each word $w_l|_{l=1}^m$
  $T$ – number of EM iterations
Output: Posteriors $p(Y_i = 1|d_i, \Theta^T)$ for each doc $d_i|_{i=1}^n$

Initialization:
for each document $d_i$ initialize $\pi_i^1$
for each word $w_l$ initialize $p_r^1(w_l) = \Omega_r \rho(w_l)$;
$p_g^1(w_l) = \frac{\Omega_g}{\rho(w_l)}$, s.t. $\Omega_r$ and $\Omega_g$ are normalization factors

Main loop:
for all $t = 1, \ldots, T$ do
  E-step:
  for each document $d_i$ compute $\alpha_i^t = p(Y_i = 1|d_i, \Theta^t)$
  for each word token $w_{ij}$ compute
      $\beta_{ij}^t = p(Z_{ij} = 1|Y_i = 1, w_{ij}, \Theta^t)$
  M-step:
  for each document $d_i$ update $\pi^{t+1} = \frac{1}{|d_i|} \sum_j \beta_{ij}^t$
  for each word $w_l$ update
      $p_r^{t+1}(w_l) = \frac{\sum_i \alpha_i^t \sum_j \delta(w_{ij} = w_l)\, \beta_{ij}^t}{\sum_i \alpha_i^t \sum_j \beta_{ij}^t}$
      $p_g^{t+1}(w_l) = \frac{N_w - \sum_i \alpha_i^t \sum_j \delta(w_{ij} = w_l)\, \beta_{ij}^t}{N - \sum_i \alpha_i^t \sum_j \beta_{ij}^t}$
```

$Y_i = 1$ corresponds to d_i being on-topic. As before, Z_{ij} decides on the topicality of a word token w_{ij}, but now *given* Y_i. Since not all words in a core document are supposed to be topical, then for each word of a core document we make a separate decision (based on Z_{ij}) whether it is sampled from $p_r(W)$ or $p_g(W)$. However, if a document does not belong to the core ($Y_i = 0$), each its word is sampled from $p_g(W)$, i.e. $p(Z_{ij} = 0|Y_i = 0) = 1$.

Inspired by Huang and Mitchell (2006), we use the Expectation-Maximization (EM) algorithm to *exactly* estimate parameters of our model from the dataset. We now describe the model parameters Θ. First, the probability of any document to belong to the core is denoted by $p(Y_i = 1) = \frac{k}{n} = p_d$ (this parameter is fixed and will not be learnt from data). Second, for each document d_i, we maintain a probability of each its word to be topical given that the document is on-topic, $p(Z_{ij} = 1|Y_i = 1) = \pi_i$ for $i = 1, \ldots, n$. Third, for each word w_l (for $k = 1...m$), we let $p(w_l|Z_l = 1) = p_r(w_l)$ and $p(w_l|Z_l = 0) = p_g(w_l)$. The overall number of parameters is $n + 2m + 1$, one of which (p_d) is preset. The dataset likelihood is then:

$$\begin{aligned} p(\mathcal{D}) &= \prod_{i=1}^{n} [p_d\, p(d_i|Y_i = 1) + (1 - p_d)p(d_i|Y_i = 0)] \\ &= \prod_{i=1}^{n} \left[p_d \prod_{j=1}^{|d_i|} [\pi_i p_r(w_{ij}) + (1 - \pi_i) p_g(w_{ij})] \right. \\ &\quad \left. + (1 - p_d) \prod_{j=1}^{|d_i|} p_g(w_{ij}) \right]. \end{aligned}$$

At each iteration t of the EM algorithm, we first perform the E-step, where we compute the posterior distribution of hidden variables $\{Y_i\}$ and $\{Z_{ij}\}$ given the current parameter values Θ^t and the data $\mathcal{D}$. Then, at the M-step, we compute the new parameter values Θ^{t+1} that maximize the model log-likelihood given Θ^t, $\mathcal{D}$ and the posterior distribution.

The initialization step is crucial for the EM algorithm. Our pilot experimentation showed that if distributions $p_r(W)$ and $p_g(W)$ are initialized as uniform, the EM performance is close to random. Therefore, we decided to initialize word probabilities using normalized ρ scores. We do not propose the optimal way to initialize π_i parameters, however, as we show later in Section 5, our LTB model appears to be quite robust to the choice of π_i.

The EM procedure is presented in Algorithm 1. For details, see Bekkerman (2008). After T iterations, we sort the documents according to α_i in decreasing order and choose the first k documents to be the core. The complexity of Algorithm 1 is linear: $O(TN)$. To avoid overfitting, we set T to be a small number: in our experiments we fix $T = 5$.

5 Experimentation

We evaluate our OCCC and LTB models on two applications: a *Web Mining* task (Section 5.1), and a *Topic Detection and Tracking (TDT)* (Allan, 2002) task (Section 5.2).

To define our evaluation criteria, let C be the constructed cluster and let C_r be its portion consisting of documents that actually belong to the core. We define precision as Prec $= |C_r|/|C|$, recall as Rec $= |C_r|/k$ and F-measure as (2 Prec Rec)/(Prec+Rec). Unless stated otherwise, in our experiments we fix $|C| = k$, such that precision equals recall and is then called *one-class clustering accuracy*, or just *accuracy*.

We applied our one-class clustering methods in four setups:

- **OCCC with the heuristic to choose** m_r (from Section 3.1).
- **OCCC with optimal** m_r**.** We *unfairly* choose the number m_r of topical words such that the resulting accuracy is maximal. This setup can be considered as the upper limit of the OCCC's performance, which can be hypothetically achieved if a better heuristic for choosing m_r is proposed.
- **LTB initialized with** $\pi_i = 0.5$ **(for each** i**).** As we show in Section 5.1 below, the LTB model demonstrates good performance with this straightforward initialization.
- **LTB initialized with** $\pi_i = p_d$**.** Quite naturally, the number of topical words in a dataset depends on the number of core documents. For example, if the core is only 10% of a dataset, it is unrealistic to assume that 50% of all words are topical. In this setup, we condition the ratio of topical words on the ratio of core documents.

We compare our methods with two existing algorithms: (a) One-Class SVM clustering[6] (Tax and Duin, 2001); (b) One-Class Rate Distortion (OC-RD) (Crammer et al., 2008). The later is considered a state-of-the-art in one-class clustering. Also, to establish the lowest baseline, we show the result of a random assignment of documents to the core $\mathcal{D}^k$.

The OC-RD algorithm is based on rate-distortion theory and expresses the one-class problem as a lossy coding of each instance into a few possible instance-dependent codewords. Each document is represented as a distribution over words, and the KL-divergence is used as a distortion function (generally, it can be any Bregman function). The algorithm also uses an "inverse temperature" parameter (denoted by β) that represents the tradeoff between compression and distortion. An annealing process is employed, in which the algorithm is applied with a sequence of increasing values of β, when initialized with the result obtained at the previous itera-

[6]We used Chih-Jen Lin's `LibSVM` with the `-s 2` parameter. We provided the core size using the `-n` parameter.

Method	WAD	TW
Random assignment	38.7%	$34.9 \pm 3.1\%$
One-class SVM	46.3%	$45.2 \pm 3.2\%$
One-class rate distortion	48.8%	$63.6 \pm 3.5\%$
OCCC with the m_r heuristic	80.2%	$61.4 \pm 4.5\%$
OCCC with optimal m	82.4%	$68.3 \pm 3.6\%$
LTB initialized with $\pi_i = 0.5$	79.8%	$65.3 \pm 7.3\%$
LTB initialized with $\pi_i = p_d$	78.3%	$68.0 \pm 5.9\%$

Table 1: One-class clustering accuracy of our OCCC and LTB models on the WAD and the TW detection tasks, as compared to OC-SVM and OC-RD. For TW, the accuracies are macro-averaged over the 26 weekly chunks, with the standard error of the mean presented after the $\pm$ sign.

tion. The outcome is a sequence of cores with decreasing sizes. The annealing process is stopped once the largest core size is equal to k.

5.1 Web appearance disambiguation

Web appearance disambiguation (WAD) is proposed by Bekkerman and McCallum (2005) as the problem of reasoning whether a particular mention of a person name in the Web refers to the person of interest or to his or her unrelated namesake. The problem is solved given *a few* names of people from one social network, where the objective is to construct a cluster of Web pages that mention names of related people, while filtering out pages that mention their unrelated namesakes.

WAD is a classic one-class clustering task, that is tackled by Bekkerman and McCallum with *simulated* one-class clustering: they use a sophisticated agglomerative/conglomerative clustering method to construct multiple clusters, out of which one cluster is then selected. They also use a simple *link structure (LS)* analysis method that matches hyperlinks of the Web pages in order to compose a cloud of pages that are close to each other in the Web graph. The authors suggest that the best performance can be achieved by a hybrid of the two approaches.

We test our models on the WAD dataset,[7] which consists of 1085 Web pages that mention 12 people names of AI researchers, such as Tom Mitchell and Leslie Kaelbling. Out of the 1085 pages, 420 are on-topic, so we apply our algorithms with $k = 420$. At a preprocessing step, we binarize document vectors and remove low frequent words (both in terms of $p(w)$ and $q(w)$). The results are summarized in the middle column of Table 1. We can see that both OCCC and LTB dramatically outperform their competitors, while showing practically indistinguishable results compared to each other. Note that when the size of the word cluster in OCCC is *unfairly* set to its optimal value, $m_r = 2200$, the OCCC method is able to gain a 2% boost. However, for obvious reasons, the optimal value of m_r may not always be obtained in practice.

[7]`http://www.cs.umass.edu/~ronb/name_disambiguation.html`

#	OCCC	LTB
1	cheyer	artificial
2	kachites	learning
3	quickreview	cs
4	adddoc	intelligence
5	aaai98	machine
6	kaelbling	edu
7	mviews	algorithms
8	mlittman	proceedings
9	hardts	computational
10	meuleau	reinforcement
11	dipasquo	papers
12	shakshuki	cmu
13	xevil	aaai
14	sangkyu	workshop
15	gorfu	kaelbling

Table 2: Most highly ranked words by OCCC and LTB, on the WAD dataset.

Table 2 lists a few most topical words according to the OCCC and LTB models. The OCCC algorithm sorts words according to their ρ scores, such that words that often occur in the dataset but rarely in the Web, are on the top of the list. These are mostly last names or login names of researchers, venues etc. The EM algorithm of LTB is the given ρ scores as an input to initialize $p_r^1(w)$ and $p_g^1(w)$, which are then updated at each M-step. In the LTB columns, words are sorted by $p_r^5(w)$. High quality of the LTB list is due to conditional dependencies in our generative model (via the Y_i nodes).

Solid lines in Figure 4 demonstrate the robustness of our models to tuning their main parameters (m_r for OCCC, and the π_i initialization for LTB). As can be seen from the left panel, OCCC shows robust performance: the accuracy above 80% is obtained when the word cluster is of any size in the 1000–3000 range. The heuristic from Section 3.1 suggests a cluster size of 1000. The LTB is even more robust: practically any value of π_i (besides the very large ones, $\pi_i \approx 1$) can be chosen.

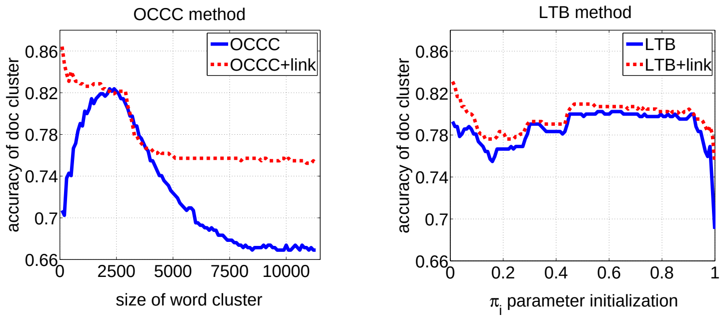


Figure 4: **Web appearance disambiguation:** (left) OCCC accuracy as a function of the word cluster size; (right) LTB accuracy over various initializations of π_i parameters. The red dotted lines show the accuracy of each method's results combined with the Link Structure model results. On the absolute scale, OCCC outperforms LTB, however LTB shows more robust behavior than OCCC.

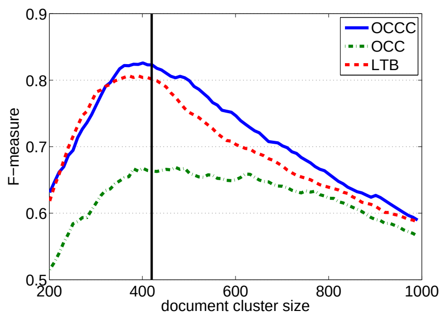


Figure 5: **Web appearance disambiguation:** F-measure as a function of document cluster size: a vertical line indicates the point where precision equals recall (and therefore equals accuracy). "OCC" refers to the OCCC model where all the words are taken as the word cluster (i.e. no word filtering is done).

To perform a fair comparison of our results with those obtained by Bekkerman and McCallum (2005), we construct hybrids of their link structure (LS) analysis model with our OCCC and LTB, as follows. First, we take their LS core cluster, which consists of 360 documents. Second, we pass over all the WAD documents in the order as they were ranked by either OCCC or LTB, and enlarge the LS core with 60 most highly ranked documents that did not occur in the LS core. In either case, we end up with a hybrid core of 420 documents.

Dotted lines in Figure 4 show accuracies of the resulting models. As the F-measure of the hybrid model proposed by Bekkerman and McCallum (2005) is 80.3%, we can see that it is significantly inferior to the results of either OCCC+LS or LTB+LS, when their parameters are set to a small value ($m_r < 3000$ for OCCC, $\pi_i < 0.06$ for LTB). Such a choice of parameter values can be explained by the fact that we need only 60 documents to expand the LS core cluster to the required size $k = 420$. When the values of m_r and π_i are small, both OCCC and LTB are able to build very small and very precise core clusters, which is exactly what we need here. The OCCC+LS hybrid is particularly successful, because it uses non-canonical words (see Table 2) to compose a clean core that almost does not overlap with the LS core. Remarkably, the OCCC+LS model obtains 86.4% accuracy with $m_r = 100$, which is the state-of-the-art result on the WAD dataset.

To answer the question how much our models are sensitive to the choice of the core size k, we computed the F-measure of both OCCC and LTB as a function of k (Figure 5). It turns out that our methods are quite robust to tuning k: choosing any value in the 300–500 range leads to good results.

5.2 Detecting the topic of the week

Real-world data rarely consists of a clean core and uniformly distributed noise. Usually, the noise has some structure, namely, it may contain coherent components. With this respect, one-class clustering can be used to detect the *largest* coherent component in a dataset, which is an integral part of many applications. In this section, we solve the problem of automatically detecting the *Topic of the Week (TW)* in a newswire stream, i.e. detecting all articles in a weekly news roundup that refer to the most broadly discussed event.

We evaluate the TW detection task on the benchmark TDT-5 dataset[8], which consists of 250 news events spread over a time period of half a year, and 9,812 documents in English, Arabic and Chinese (translated to English), annotated by their relationship to those events.[9] The largest event in TDT-5 dataset (#55106, titled *"Bombing in Riyadh, Saudi Arabia"*) has 1,144 documents, while 66 out of the 250 events have only one document each. We split the dataset to 26 weekly chunks (to have 26 full

[8]`http://projects.ldc.upenn.edu/TDT5/`

[9]We take into account only labeled documents, while ignoring unlabeled documents that can be found in the TDT-5 data.

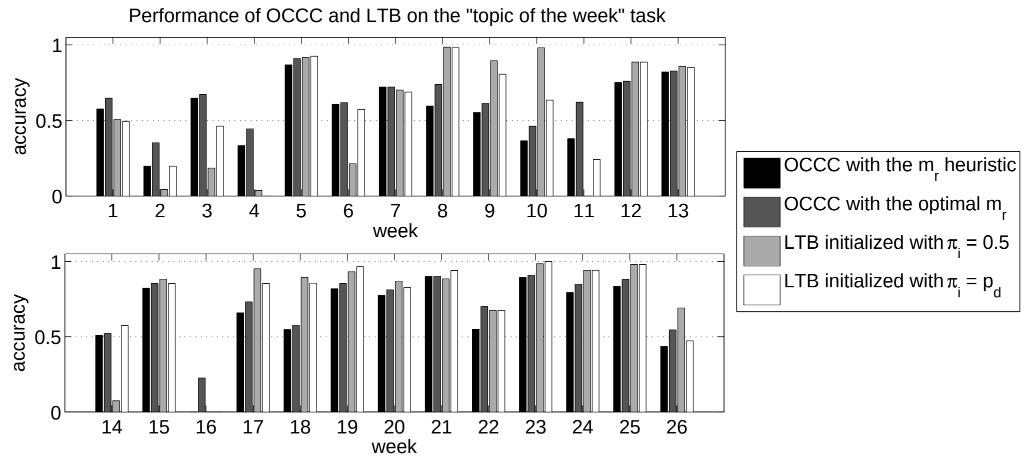


Figure 6: **"Topic of the week" detection task:** Accuracies of two OCCC methods and two LTB methods.

weeks, we delete all the documents dated with the last day in the dataset, which decreases the dataset's size to 9,781 documents). Each chunk contains from 138 to 1292 documents.

The one-class clustering accuracies, macro-averaged over the 26 weekly chunks, are presented in the right column of Table 1. As we can see, both LTB models, as well as OCCC with the optimal m_r, outperform our baselines. Interestingly, even the optimal choice of m_r does not lead OCCC to significantly superior results while compared with LTB. The dataset-dependent initialization of LTB's π_i parameters ($\pi_i = p_d$) appears to be preferable over the dataset-independent one ($\pi_i = 0.5$).

Accuracies *per week* are shown in Figure 6. These results reveal two interesting observations. First, OCCC tends to outperform LTB only on data chunks where the results are quite low in general (less than 60% accuracy). Specifically, on weeks 2, 4, 11, and 16 the LTB models show extremely poor performance. While investigating this phenomenon, we discovered that in two of the four cases LTB was able to construct very clean core clusters, however, those clusters corresponded to the *second* largest topic, while we evaluate our methods on the first largest topic.[10] Second, the (completely unsupervised) LTB model can obtain very good results on some of the data chunks. For example, on weeks 5, 8, 19, 21, 23, 24, and 25 the LTB's accuracy is above 90%, with a striking 100% on week-23.

6 Conclusion

We have developed the theory and proposed practical methods for one-class clustering in the text domain. The proposed algorithms are very simple, very efficient and still surprisingly effective. More sophisticated algorithms (e.g. an *iterative*[11] version of OCCC) are emerging.

7 Acknowledgements

We thank Erik Learned-Miller for the inspiration on this project. We also thank Gunjan Gupta, James Allan, and Fernando Diaz for fruitful discussions. 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

J. Allan, editor. 2002. *Topic detection and tracking: event-based information organization.* Kluwer Academic Publishers.

[10]For example, on the week-4 data, topic #55077 (*"River ferry sinks on Bangladeshi river"*) was discovered by LTB as the largest and most coherent one. However, in that dataset, topic #55077 is represented by 20 documents, while topic #55063 (*"SARS Quarantined medics in Taiwan protest"*) is represented by 27 documents, such that topic #55077 is in fact the second largest one.

[11]See, e.g., El-Yaniv and Souroujon (2001)

R. Bekkerman and A. McCallum. 2005. Disambiguating web appearances of people in a social network. In *Proceedings of WWW-05, the 14th International World Wide Web Conference*.

R. Bekkerman. 2008. *Combinatorial Markov Random Fields and their Applications to Information Organization*. Ph.D. thesis, University of Massachusetts at Amherst.

K. Crammer and G. Chechik. 2004. A needle in a haystack: local one-class optimization. In *Proceedings of the 21st International Conference on Machine Learning*.

K. Crammer, P. Talukdar, and F. Pereira. 2008. A rate-distortion one-class model and its applications to clustering. In *Proceedings of the 25st International Conference on Machine Learning*.

N. Cristianini and J. Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

R. El-Yaniv and O. Souroujon. 2001. Iterative double clustering for unsupervised and semi-supervised learning. In *Advances in Neural Information Processing Systems (NIPS-14)*.

G. Gupta and J. Ghosh. 2005. Robust one-class clustering using hybrid global and local search. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 273–280.

Y. Huang and T. Mitchell. 2006. Text clustering with extended user feedback. In *Proceedings of the 29th annual international ACM SIGIR conference*, pages 413–420.

G. Lebanon. 2005. *Riemannian Geometry and Statistical Machine Learning*. Ph.D. thesis, CMU.

B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson. 2001. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471.

N. Slonim and N. Tishby. 2000. Document clustering using word clusters via the information bottleneck method. In *Proceedings of the 23rd annual international ACM SIGIR conference*, pages 208–215.

T. Tao and C. Zhai. 2004. A two-stage mixture model for pseudo feedback. In *Proceedings of the 27th annual international ACM SIGIR conference*, pages 486–487.

D. M. J. Tax and R. P. W. Duin. 2001. Outliers and data descriptions. In *Proceedings of the 7th Annual Conference of the Advanced School for Computing and Imaging*, pages 234–241.

Y. Zhou and W. B. Croft. 2007. Query performance prediction in web search environments. In *Proceedings of the 30th Annual International ACM SIGIR Conference*.

Refining Generative Language Models using Discriminative Learning

Ben Sandbank
Blavatnik School of Computer Science
Tel-Aviv University
Tel-Aviv 69978, Israel
sandban@post.tau.ac.il

Abstract

We propose a new approach to language modeling which utilizes discriminative learning methods. Our approach is an iterative one: starting with an initial language model, in each iteration we generate 'false' sentences from the current model, and then train a classifier to discriminate between them and sentences from the training corpus. To the extent that this succeeds, the classifier is incorporated into the model by lowering the probability of sentences classified as false, and the process is repeated. We demonstrate the effectiveness of this approach on a natural language corpus and show it provides an 11.4% improvement in perplexity over a modified kneser-ney smoothed trigram.

1 Introduction

Language modeling is a fundamental task in natural language processing and is routinely employed in a wide range of applications, such as speech recognition, machine translation, etc'. Traditionally, a language model is a probabilistic model which assigns a probability value to a sentence or a sequence of words. We refer to these as *generative language models*. A very popular example of a generative language model is the n-gram, which conditions the probability of the next word on the previous *(n-1)*-words.

Although simple and widely-applicable, it has proven difficult to allow n-grams, and other forms of generative language models as well, to take advantage of non-local and overlapping features.[1] These sorts of features, however, pose no problem for standard discriminative learning methods, e.g. large-margin classifiers. For this reason, a new class of language model, the *discriminative language model*, has been proposed recently to augment generative language models (Gao et al., 2005; Roark et al., 2007). Instead of providing probability values, discriminative language models directly classify sentences as either correct or incorrect, where the definition of correctness depends on the application (e.g. grammatical / ungrammatical, correct translation / incorrect translation, etc').

Discriminative learning methods require negative samples. Given that the corpora used for training language models contain only real sentences, i.e. positive samples, obtaining these can be problematic. In most work on discriminative language modeling this was not a major issue as the work was concerned with specific applications, and these provided a natural definition of negative samples. For instance, (Roark et al., 2007) proposed a discriminative language model for a speech recognition task. Given an acoustic sequence, a baseline recognizer was used to generate a set of possible transcriptions. The correct transcription was taken as a positive sample, while the rest were taken as negative samples. More recently, however, Okanohara and Tsujii (2007) showed that a

[1] Conditional maximum entropy models (Rosenfeld, 1996) provide somewhat of a counter-example, but there, too, many kinds of global and non-local features are difficult to use (Rosenfeld, 1997).

discriminative language model can be trained independently of a specific application by using a generative language model to obtain the negative samples. Using a non-linear large-margin learning algorithm, they successfully trained a classifier to discriminate real sentences from sentences generated by a trigram.

In this paper we extend this line of work to study the extent to which discriminative learning methods can lead to better *generative language models per-se*. The basic intuition is the following: if a classifier can be used to discriminate real sentences from 'false' sentences generated by a language model, then it can also be used to improve that language model by taking probability mass away from sentences classified as false and transferring it to sentences classified as real. If the resulting language model can be efficiently sampled from, then this process can be repeated, until generated sentences can no longer be distinguished from real ones.

The remainder of the paper is structured as follows: In the next section we formally develop this intuition, providing a quick overview of the whole-sentence maximum-entropy model and of self-supervised boosting, two previous works on which we rely. We also present the method we use for sampling from the current model, which for the present work is far more efficient than the classical Gibbs sampling. Our experimental results are presented in section 3, and section 4 concludes with a discussion and a future outlook.

2 Learning Framework

2.1 Whole-sentence maximum-entropy model

The vast majority of statistical language models estimate the probability of a given sentence as a product of conditional probabilities via the chain rule:

$$\mathrm{P}(s) \overset{def}{=} \mathrm{P}(w_1 ... w_n) \overset{def}{=} \prod_{i=1}^{n} \mathrm{P}(w_i \mid h_i) \qquad (1)$$

where $h_i \overset{def}{=} w_1 ... w_{i-1}$ is called the *history* of the word w_i. Most work on language modeling therefore is directed at the estimation of $P(w_i \mid h_i)$. While this is theoretically correct, it makes it difficult to incorporate global information about the sentence into the model, e.g. length, grammaticality, etc'. For this reason, the whole-sentence maximum-entropy model was proposed in (Rosenfeld, 1997). In the WSME model the probability of a sentence is defined directly as:

$$P(s) = \frac{1}{Z} P_0(s) \cdot \exp(\sum_i \lambda_i f_i(s)) \qquad (2)$$

Where $P_0(s)$ is some baseline model, $Z \overset{def}{=} \sum_s P_0(s) \cdot \exp(\sum_i \lambda_i f_i(s))$ is a normalization constant and the $\{f_i\}$'s are *features* encoding some information about the sentence. Most generally, a feature is a function from the set of word sequences to *R*, the set of real numbers. However, in most applications, as in our work, the features are taken to be binary. Lastly, the $\{\lambda_i\}$'s are real coefficients encoding the relative importance of their corresponding features. In the WSME framework the set of features $\{f_i\}$ is given ahead of training by the modeler, and learning consists of estimating the coefficients $\{\lambda_i\}$. This is done by stipulating the constraints

$$E_p(f_i) = E_{\tilde{p}}(f_i) \overset{def}{=} \frac{1}{N} \sum_{j=1}^{N} f_i(s_j) \qquad (3)$$

where $\tilde{p}$ is the empirical distribution defined by the training set $\{s_1, ... s_N\}$.[2] If these constraints are consistent then there is a unique solution in $\{\lambda_i\}$ that satisfies them. This solution is guaranteed to be the one closest to P_0 in the Kullback-Leblier sense among all solutions satisfying (3). It is also guaranteed to be the maximum likelihood solution for the exponential family. For more details, see (Chen and Rosenfeld, 1999a).

2.2 Self-supervised boosting

A different approach to learning the same sort of model as in (2) was proposed in (Welling et al., 2003). Here, instead of having all the features pre-given, they are *learned* one at a time along with their corresponding coefficients. Welling et al. show that adding a new feature to (2) can be

[2] Sometimes a smoothed version of (3) is used instead (e.g. Chen and Rosenfeld, 1999b).

interpreted as gradient ascent on the log-likelihood function, and show that the optimal feature is the one that best discriminates real data from data sampled from the current model. To see this, let

$$E(s) = \ln(P_0(s)) + \sum_i \lambda_i f_i(s) \tag{4}$$

denote the *energy* associated with sentence s.[3] Equation (2) can now be rewritten as -

$$P(s) = \frac{1}{Z}\exp(E(s)) \tag{5}$$

where Z is a normalization constant as before. The derivative of the log-likelihood with respect to an arbitrary parameter β is then –

$$\frac{\partial L}{\partial \beta} = -\frac{1}{N}\sum_{i=1}^{N}\frac{\partial E(s_i)}{\partial \beta} + \sum_{s \in S} P(s)\frac{\partial E(s)}{\partial \beta} \tag{6}$$

where $\{s_1, \ldots s_N\}$ is once again the training corpus, and the second sum runs over the set of all word sequences.

Now, suppose we change the energy function by adding an infinitesimal multiple of a new feature f^*. The log-likelihood after adding the feature can be approximated by –

$$L(E(s) + \varepsilon f^*(s)) \approx L(E(s)) + \varepsilon \frac{\partial L}{\partial \varepsilon} \tag{7}$$

where the derivative of L is taken at $\varepsilon = 0$. Because the optimal feature is the one that maximizes the increase in log-likelihood, we are searching for a feature that maximizes this derivative. Using equation (6) and noting that $\frac{\partial E}{\partial \varepsilon} = f^*$ we have –

$$\frac{\partial L}{\partial \varepsilon} = -\frac{1}{N}\sum_{i=1}^{N} f^*(s_i) + \sum_{s \in S} P(s) f^*(s) \tag{8}$$

This expression cannot be computed in practice, because the set of all word sequences S is infinite. The second term however can approximated using samples $\{u_i\}$ from the current model –

$$\frac{\partial L}{\partial \varepsilon} \approx -\frac{1}{N}\sum_{i=1}^{N} f^*(s_i) + \frac{1}{N}\sum_{i=1}^{N} f^*(u_i) \tag{9}$$

[3] In (Welling et al., 2003) the term for P_0 does not appear, which is equivalent to taking the uniform distribution as the baseline model.

In other words, given a set of N samples $\{u_i\}$ from the model, the optimal feature to add is one that gives high scores to sampled sentences and low ones to real sentences. By labeling real sentences with 0 and sampled sentences with 1, the task of learning the feature translates into the task of training a classifier to discriminate between these two classes of sentences.

In the remainder of the paper we will use *feature* and *classifier* interchangeably.

2.3 Rejection sampling

Self-supervised boosting was presented as a general method for density estimation, and was not tested in the context of language modeling. Rather, Welling at al. demonstrated its effectiveness in modeling hand-written digits and on synthetic data. In both cases essentially linear classifiers were used as features. As these are computationally very efficient, the authors could use a variant of Gibbs sampling for generating negative samples. Unfortunately, as shown in (Okanohara and Tsujii, 2007), with the represetation of sentences that we use, linear classifiers cannot discriminate real sentences from sentences sampled from a trigram, which is the model we use as a baseline, so here we resort to a non-linear large-margin classifier (see section 3 for details). While large-margin classifiers consistently out-perform other learning algorithms in many NLP tasks, their non-linear variations are also notoriously slow when it comes to computing their decision function – taking time that can be linear in the size of their training data. This means that MCMC techniques like Gibbs sampling quickly become intractable, even for small corpora, as they require performing very large numbers of classifications. For this reason we use a different sampling scheme which we refer to as *rejection sampling*. This allows us to sample from the true model distribution while requiring a drastically smaller number of classifications, as long as the current model isn't too far removed from the baseline.

We will start by describing the sampling process, and then show that the probability distribution it samples from has the form of equation (2). To sample a sentence from the current model, we generate one from the baseline model, and then pass it through each of the classifiers in the model. If a given classifier classifies the sentence as a model sentence, then it is rejected with a certain probability associated with this classifier. Only if a sentence is accepted by all classifiers is it taken as a sample sentence. Otherwise, the sampling process is restarted.

Let us derive an expression for the probability of a sentence *s* generated in this manner. To simplify notation, assume that at this point we added but a single feature *f* to the baseline model P_0, and let p_{rej} stand for the rejection probability associated with it. Furthermore, let p_- stand for the accuracy of *f* in classifying sentences sampled from P_0 (negative samples). Formally,

$$p_- = E_{P_0}(f) \tag{10}$$

First let's assume that $f(s)=1$. The probability for generating s is a sum of the probabilities of two disjoint outcomes – the probability of generating s as the first sentence and having it survive the rejection, plus the probability of generating in the first iteration some sentence s' such that $f(s')=1$, rejecting that, and then generating s in one of the subsequent iterations. Formally, this means that –

$$P_1(s) = (1-p_{rej})P_0(s) + p_- p_{rej} P_1(s) \tag{11}$$

Rearranging, we have –

$$P_1(s) = \frac{1-p_{rej}}{1-p_- p_{rej}} P_0(s) \tag{12}$$

Similarly, the probability for a sentence s for which $f(s)=0$ is the probability of generating s as the first sentence, plus the probability of generating some other sentence s' for which $f(s')=1$, rejecting it, and then generating s in a future iteration. Formally,

$$P_1(s) = P_0(s) + p_- p_{rej} P_1(s) \tag{13}$$

and hence –

$$P_1(s) = \frac{1}{1-p_- p_{rej}} P_0(s) \tag{14}$$

Letting $Z = 1 - p_- p_{rej}$, and letting $\lambda = \ln(1-p_{rej})$, we have, for all s –

$$P_1(s) = \frac{1}{Z} P_0(s) \cdot \exp(\lambda \cdot f(s)) \tag{15}$$

This process can be trivially generalized for N features. Let –

$$p_-^i = E_{P_{i-1}}(f_i) \tag{16}$$

stand for f_i's accuracy in classifying sentences generated from P_{i-1}, and let p_{rej}^i be the rejection probability associated with the i'th feature. Sampling from the model then proceeds by sampling a sentence *s* from P_0. For each $1 \le i \le N$, in order, if $f_i(s)=1$, then we attempt to reject *s* with probability p_{rej}^i. If *s* survives all the rejection attempts, it is returned as the next sample. Using similar arguments as before it's possible to show that if we take $\lambda_i = \ln(1-p_{rej}^i)$ and –

$$Z = \prod_{i=1}^{N} (1 - p_-^i p_{rej}^i) \tag{17}$$

then the probability of a sentence *s* sampled by this process is given by equation (2). Conversely, this shows that rejection sampling can be used for obtaining negative samples from the model given in (2) by taking $p_{rej}^i = 1 - \exp(\lambda_i)$, as long as $0 < \exp(\lambda_i) \le 1$. In section 3 we show that in our experimental setup, rejection sampling brings about enormous savings in the number of classifications necessary during training, as compared with Gibbs sampling.

2.4 Adding a new feature

Given the current model P_i and a new feature f_{i+1}, we wish to find the optimal λ_{i+1}, or equivalently its optimal rejection probability p_{rej}^{i+1}. In the WSME framework, the weights of the features are set in such a way that the expected value of the features on sentences sampled from the model equals their expected value on real sentences. A possible way to set the weight of a new feature is therefore to set p_{rej}^{i+1} such that the constraint:

$$E_{P_{i+1}}(f_{i+1}) = E_{\tilde{p}}(f_{i+1}) \tag{18}$$

is satistfied, where $\tilde{p}$ is once again the empirical distribution defined by the training set. Intuitively, this means that the new feature could no longer be used to discriminate between sentences sampled from P_{i+1} and real sentences. However, setting

p_{rej}^{i+1} in this manner may violate the constraints (18) associated with the features already existing in the model, thus hampering the model's performance. Therefore, we set the new feature's rejection probability by directly searching for the one that minimizes an estimate of P_{i+1}'s perplexity on a set of held out real sentences. To do this, we first sample a new set of sentences from P_i, independently of the set that was used for training f_{i+1}, and use it to estimate p_{-}^{i+1}. For any arbitrarily determined p_{rej}^{i+1}, this enables us to calculate an estimate for the normalization constant Z (equation 17), and therefore an estimate for P_{i+1}. We do this for a range of possible values for p_{rej}^{i+1} and pick the one that leads to the largest reduction in perplexity on the held out data.[4]

3 Experimental work

We tested our approach on the ATIS natural language corpus (Hemphill et al., 1990). We split the corpus into a training set of 11,000 sentences, a held-out set containing 1,045 sentences, and a test set containing 1,000 sentences which were reserved for measuring perplexity. The corpus was pre-processed so that every word appearing less than three times was replaced by a special UNK symbol. The resulting lexicon contained 603 word types.

Our learning framework leaves open a number of design choices:

1. **Baseline language model**: For P_0 we used a trigram with modified kneser-ney smoothing [Chen and Goodman, 1998], which is still considered one of the best smoothing methods for n-gram language models.

2. **Sentence representation**: Each sentence was represented as the collection of unigrams, bigrams and trigrams it contained. A coordinate was reserved for each such n-gram which appeared in the data, whether real or sampled. The value of the n'th coordinate in the vector representation of sentence s was set to the number of times the corresponding n-gram appeared in s.

3. **Type of classifiers**: For our features we used large-margin classifiers trained using the online algorithm described in (Crammer et al., 2006). The code for the classifier was generously provided by Daisuke Okanohara. This code was extensively optimized to take advantage of the very sparse sentence representation described above. As shown in (Okanohara and Tsujii, 2007), using this representation, a linear classifier cannot distinguish sentences sampled from a trigram and real sentences. Therefore, we used a 3rd order polynomial kernel, which was found to give good results. No special effort was otherwise made in order to optimize the parameters of the classifiers.

4. **Stopping criterion**: The process of adding features to the model was continued until the classification performance of the next feature was within 2% of chance performance.

We refer to the language model obtained by this approach as the *boosted* model to distinguish it from the baseline model. To estimate the boosted model's perplexity we needed to estimate the normalization constant Z in equation (2). Since this constant is equal to $E_{P_0}(\exp(\sum_i \lambda_i f_i))$ it can be estimated from a large-enough sample from P_0. We used 10,000,000 sentences generated from the baseline trigram and took the upper bound of the 95% confidence interval of the sample mean as an upper bound for Z. This means the perplexity estimates we report are upper bounds for the real model perplexity with 95% confidence.[5]

The algorithm converged after 21 features were added to the model. Figure 1 presents the model's perplexity on the test set estimated after each iteration. The perplexity of the final model is 9.02. In comparison, the perplexity of the modified kneser-ney smoothed trigram on this corpus is 10.18. This is an 11.4% improvement relative to the baseline model.

[4] Interestingly, in practice both methods result in near identical rejection probabilities, within a precision of 0.0001. This indicates that satisfying the constraint (18) for the new feature is more important, in terms of perplexity, than preserving the constraints of the previous features, insofar as those get violated.

[5] Alternatively we could have used our estimate for P_N(s) described in section 2.4. A large sample of sentences would still be necessary though, to get a good estimate for equation (16).

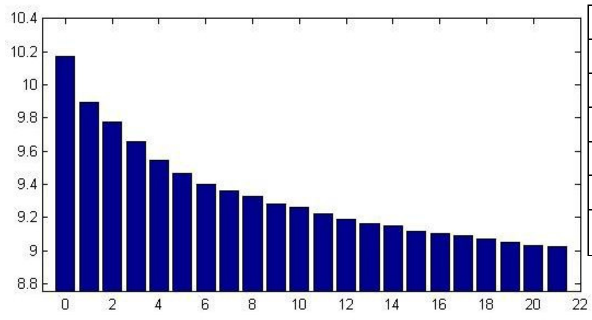


Figure 1. Model perplexity during training. The x-axis denotes the number of features added to the model. The final perplexity after 21 features is 9.02.

please list costs in at pittsburgh
what type of airplane is have an early morning
what types of aircraft is that a meal
what not nineteen forty two
between boston and atlanta on august fifteenth
which airlines fly american flying on
what is the flight leaving pittsburgh after six p m

Table 1. A sample of sentences generated by the baseline model, a trigram smoothed with modified kneser-ney smoothing.

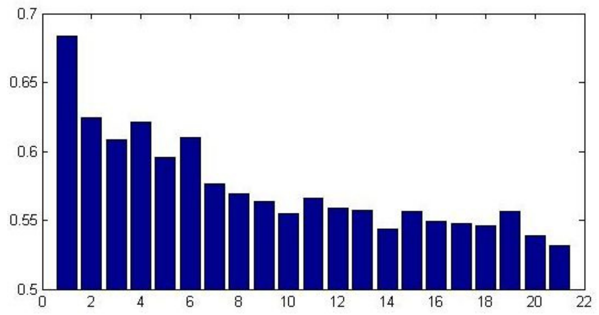


Figure 2. Classifier accuracy during training, assessed on held-out data. 0.5 signifies chance performance.

what is the cost of flight d l three seventeen sixty five
what time does flight at eight thirty eight a m and six p m
what does fare code q w mean
what kind of aircraft will i be flying on
flights from philadelphia on saturday
what is the fare for flight two nine six
what is the cost of coach transcontinental flight u a three oh two from denver to san francisco

Table 2. A sample of sentences generated by the final model

Figure 2 shows the accuracy of the trained features on held-out data. The held-out data was composed of equal parts real and model sentences, so 50% accuracy is chance performance. As might have been expected, the classifiers start out with a relatively high accuracy of 68%, which dwindles down to little over 50% as more features are added to the model. Not surprisingly, there is a strong correlation between the accuracy of a feature and the reduction in perplexity it engenders (spearman correlation coefficient r=0.89, $p<10^{-5}$.)

In tables 1 and 2 we show a representative sample of sentences from the baseline model and from the final model. As the baseline model is a trigram, it cannot capture dependencies that span a range longer than two words. Hence sentences that start out seemingly in one topic and then veer off to another are common. The global information available to the features used by the boosted model greatly reduces this phenomenon. To get a quantitative sense of this, we generated 200 sentences from each model and submitted them for grammaticality testing by a proficient (though non-native) English speaker. Of the trigram-generated sentences, 86 were deemed grammatical (43%), while of those generated by the boosted model 132 were grammatical (66%). This difference is statistically significant with $p<10^{-5}$.

Finally, let us quantify the computational savings obtained from using rejection sampling. Let |V| stand for the lexicon size (here |V|=603) and |L| for the average sentence length (|L|=14). In Gibbs sampling, a sentence is sampled by starting out with a random sequence of words. For each word position, the current word is replaced with each word in the lexicon, and the probability of the resulting sentence is calculated. Then one of the words is randomly selected for this position in proportion to the calculated probabilities. The sentence has to be scanned in this manner several times for the sample to approximate the model distribution. Assuming we perform only 3 scans for each sentence, Gibbs sampling would have thus required us to classify $3|V||L| \approx 25,000$ sentences *per sampled sentence*. Given that in each iteration we generate 12,045 sentences, and that in the *n'th* iteration each sentence has to be classified by *n* features, this gives a total of roughly

$7 \cdot 10^{10}$ classifications after 21 iterations. In contrast, using rejection sampling, we used only $6.7 \cdot 10^{7}$ classifications in total – a difference of over three orders of magnitude.

4 Discussion

In this work we presented a method that enables using discriminative learning methods for refining generative language models. Utilizing large-margin classifiers that are trained to discriminate real sentences from model sentences we showed that sizeable improvements in perplexity over a state-of-the-art smoothed trigram are possible.

Our method bears some similarity to the recently developed *Contrastive Estimation* method (Smith and Eisner, 2004). Contrastive estimation (CE) was proposed as a means for training log-linear probabilistic models. As all training methods, contrastive estimation pushes probability mass unto positive samples. Unlike other methods, CE takes this probability mass from the 'neighborhood' of each positive sample. For example, given a real sentence *s*, CE might give it more probability by taking away probability from similar sentences which are likely to be ungrammatical, for instance sentences that are formed by taking *s* and switching the order of two adjacent words in it. This is intuitively similar to our approach – effectively, our model gives probability mass to positive samples, taking it away from sentences classified as model sentences. A major difference between the two approaches, however, is that in CE the definition of the sentence's neighborhood must be specified in advance by the modeler. In our work, the 'neighborhood' is determined automatically and dynamically as learning proceeds, according to the capabilities of the classifiers used.

The sentence representation we chose for this work is rather simple, and was intended primarily to demonstrate the efficacy of our approach. In future work we plan to experiment with richer representations, e.g. including long-range n-grams (Rosenfeld, 1996), class n-grams (Brown et al., 1992), grammatical features (Amaya and Benedy, 2001), etc'.

The main computational bottleneck in our approach is the generation of negative samples from the current model. Rejection sampling allowed us to use computationally intensive classifiers as our features by reducing the number of classifications that had to be performed during the sampling process. However, if the boosted model strays too far from the baseline P_0, these savings will be negated by the very large sentence rejection probabilities that will ensue. This is likely to be the case when richer representations as suggested above are used, necessitating a return to Gibbs sampling. Therefore, in future work we plan to experiment with classifiers whose decision function is cheaper to compute, such as neural networks and decision trees. Another possible direction would be using the recently proposed Deep Belief Network formalism (Hinton et al., 2006). DBNs utilize semi-linear features which are stacked recursively and thus very efficiently model non-linearities in their data. These have been used in the past for language modeling (Mnih and Hinton, 2007), but not within the whole-sentence framework.

Acknowledgements

We would like to thank Daisuke Okanohara for his generosity in providing the code for the large-margin classifier. The author is supported by the Yeshaya Horowitz association through the center of Complexity Science.

References

Fredy Amaya and Jose Miguel Benedi, 2001, Improvement of a whole sentence maximum entropy model using grammatical features. In *Proceedings of the 39th Annual Meeting of the Association of Computational Linguistics.*

Peter. F. Brown, Vincent. J. Della Pietra, Peter. V. deSouza, Jenifer. C. Lai and Robert. L. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, Dec. 1992.

Stanley F. Chen and Joshua. Goodman. 1998. An empirical study of smoothing techniques for language modeling. *Technical Report TR–10–98*, Center for Research in Computing Technology, Harvard University

Stanley F. Chen and Ronald Rosenfeld. 1999a. Efficient sampling and feature selection in whole sentence maximum entropy language models. In *Proceedings of ICASSP'99. IEEE.*

Stanley F. Chen and Ronald Rosenfeld. 1999b. A Gaussian prior for smoothing maximum entropy models. *Technical Report CMUCS-99-108*, Carnegie Mellon University.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research.*

Jianfeng Gao, Hao Yu, Wei Yuan, and Peng Xu. 2005. Minimum sample risk methods for language modeling. In *Proc. of HLT/EMNLP.*

Charles T. Hemphill, John J. Godfrey and George R. Doddington. 1990. The ATIS Spoken Language Systems Pilot Corpus. *The workshop on speech and natural language*, Morgan Kaufmann.

Geoffrey E. Hinton, Simon Osindero and Yee-Whye Teh, 2006. A fast learning algorithm for deep belief nets, *Neural Computation*, 18(7):1527–1554.

Andriy Mnih and Geoffrey E. Hinton. 2007. Three new graphical models for statistical language modeling. In *Proceedings of the 24th international conference on Machine Learning.*

Daisuke Okanohara and Jun'ichi Tsujii. 2007. A discriminative language model with pseudo-negative samples. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics.*

Brian Roark, Murat Saraclar, and Michael Collins. 2007. Discriminative n-gram language modeling. *Computer Speech and Language*, 21(2):373–392.

Ronald Rosenfeld. 1996. A maximum entropy approach to adaptive statistical language modeling. *Computer speech and language*, 10:187–228

Ronald Rosenfeld. 1997. A whole sentence maximum entropy language model. In *Proc. of the IEEE Workshop on Automatic Speech Recognition and Understanding.*

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the Association of Computational Linguistics*.

Max Welling., Richard Zemel and Geoffrey E. Hinton. 2003. Self-Supervised Boosting. *Advances in Neural Information Processing Systems*, 15, MIT Press, Cambridge, MA

Discriminative Learning of Selectional Preference from Unlabeled Text

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

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

Randy Goebel
Department of Computing Science
University of Alberta
Edmonton, Alberta
Canada, T6G 2E8
goebel@cs.ualberta.ca

Abstract

We present a discriminative method for learning selectional preferences from unlabeled text. Positive examples are taken from observed predicate-argument pairs, while negatives are constructed from unobserved combinations. We train a Support Vector Machine classifier to distinguish the positive from the negative instances. We show how to partition the examples for efficient training with 57 thousand features and 6.5 million training instances. The model outperforms other recent approaches, achieving excellent correlation with human plausibility judgments. Compared to Mutual Information, it identifies 66% more verb-object pairs in unseen text, and resolves 37% more pronouns correctly in a pronoun resolution experiment.

1 Introduction

Selectional preferences (SPs) tell us which arguments are plausible for a particular predicate. For example, Table 2 (Section 4.4) lists plausible and implausible direct objects (arguments) for particular verbs (predicates). SPs can help resolve syntactic, word sense, and reference ambiguity (Clark and Weir, 2002), and so gathering them has received a lot of attention in the NLP community.

One way to determine SPs is from co-occurrences of predicates and arguments in text. Unfortunately, no matter how much text we use, many acceptable pairs will be missing. Bikel (2004) found that only 1.49% of the bilexical dependencies considered by Collins' parser during decoding were observed during training. In our parsed corpus (Section 4.1), for example, we find *eat* with *nachos*, *burritos*, and *tacos*, but not with the equally tasty *quesadillas*, *chimichangas*, or *tostadas*. Rather than solely relying on co-occurrence counts, we would like to use them to generalize to unseen pairs.

In particular, we would like to exploit a number of arbitrary and potentially overlapping properties of predicates and arguments when we assign SPs. We do this by representing these properties as features in a linear classifier, and training the weights using discriminative learning. Positive examples are taken from observed predicate-argument pairs, while pseudo-negatives are constructed from unobserved combinations. We train a Support Vector Machine (SVM) classifier to distinguish the positives from the negatives. We refer to our model's scores as Discriminative Selectional Preference (DSP). By creating training vectors automatically, DSP enjoys all the advantages of supervised learning, but without the need for manual annotation of examples.

We evaluate DSP on the task of assigning verb-object selectional preference. We encode a noun's textual distribution as feature information. The learned feature weights are linguistically interesting, yielding high-quality similar-word lists as latent information. Despite its representational power, DSP scales to real-world data sizes: examples are partitioned by predicate, and a separate SVM is trained for each partition. This allows us to efficiently learn with over 57 thousand features and 6.5 million examples. DSP outperforms recently proposed alternatives in a range of experiments, and better correlates with human plausibility judgments. It also shows strong gains over a Mutual Information-based co-occurrence model on two tasks: identifying objects of verbs in an unseen corpus and finding pronominal antecedents in coreference data.

2 Related Work

Most approaches to SPs generalize from observed predicate-argument pairs to semantically similar ones by modeling the semantic class of the argument, following Resnik (1996). For example, we might have a class *Mexican Food* and learn that the entire class is suitable for eating. Usually, the classes are from WordNet (Miller et al., 1990), although they can also be inferred from clustering (Rooth et al., 1999). Brockmann and Lapata (2003) compare a number of WordNet-based approaches, including Resnik (1996), Li and Abe (1998), and Clark and Weir (2002), and found that the more sophisticated class-based approaches do not always outperform simple frequency-based models.

Another line of research generalizes using similar words. Suppose we are calculating the probability of a particular noun, n, occurring as the object argument of a given verbal predicate, v. Let $\text{Pr}(n|v)$ be the empirical maximum-likelihood estimate from observed text. Dagan et al. (1999) define the similarity-weighted probability, Pr_{SIM}, to be:

$$\text{Pr}_{\text{SIM}}(n|v) = \sum_{v' \in \text{SIMS}(v)} \textit{Sim}(v', v)\text{Pr}(n|v') \quad (1)$$

where $\textit{Sim}(v', v)$ returns a real-valued similarity between two verbs v' and v (normalized over all pair similarities in the sum). In contrast, Erk (2007) generalizes by substituting similar *arguments*, while Wang et al. (2005) use the cross-product of similar pairs. One key issue is how to define the set of similar words, $\text{SIMS}(w)$. Erk (2007) compared a number of techniques for creating similar-word sets and found that both the Jaccard coefficient and Lin (1998a)'s information-theoretic metric work best. Similarity-smoothed models are simple to compute, potentially adaptable to new domains, and require no manually-compiled resources such as WordNet.

Selectional Preferences have also been a recent focus of researchers investigating the learning of paraphrases and inference rules (Pantel et al., 2007; Roberto et al., 2007). Inferences such as "[*X* wins *Y*] ⇒ [*X* plays *Y*]" are only valid for certain arguments *X* and *Y*. We follow Pantel et al. (2007) in using automatically-extracted semantic classes to help characterize plausible arguments.

Discriminative techniques are widely used in NLP and have been applied to the related tasks of word prediction and language modeling. Even-Zohar and Roth (2000) use a classifier to predict the most likely word to fill a position in a sentence (in their experiments: a verb) from a set of candidates (sets of verbs), by inspecting the context of the target token (e.g., the presence or absence of a particular nearby word in the sentence). This approach can therefore learn which specific arguments occur with a particular predicate. In comparison, our features are second-order: we learn what *kinds* of arguments occur with a predicate by encoding features of the arguments. Recent distributed and latent-variable models also represent words with feature vectors (Bengio et al., 2003; Blitzer et al., 2005). Many of these approaches learn both the feature weights and the feature representation. Vectors must be kept low-dimensional for tractability, while learning and inference on larger scales is impractical. By partitioning our examples by predicate, we can efficiently use high-dimensional, sparse vectors.

Our technique of generating negative examples is similar to the approach of Okanohara and Tsujii (2007). They learn a classifier to disambiguate actual sentences from pseudo-negative examples sampled from an N-gram language model. Smith and Eisner (2005) also automatically generate negative examples. They perturb their input sequence (e.g. the sentence word order) to create a neighborhood of *implicit* negative evidence. We create negatives by substitution rather than perturbation, and use corpus-wide statistics to choose our negative instances.

3 Methodology

3.1 Creating Examples

To learn a discriminative model of selectional preference, we create positive and negative training examples automatically from raw text. To create the positives, we automatically parse a large corpus, and then extract the predicate-argument pairs that have a statistical association in this data. We measure this association using pointwise Mutual Information (MI) (Church and Hanks, 1990). The MI between a verb predicate, v, and its object argument, n, is:

$$\mathrm{MI}(v,n) = \log\frac{\Pr(v,n)}{\Pr(v)\Pr(n)} = \log\frac{\Pr(n|v)}{\Pr(n)} \quad (2)$$

If MI>0, the probability v and n occur together is greater than if they were independently distributed.

We create sets of positive and negative examples separately for each predicate, v. First, we extract all pairs where $\mathrm{MI}(v,n)>\tau$ as positives. For each positive, we create pseudo-negative examples, (v,n'), by pairing v with a new argument, n', that either has MI below the threshold or did not occur with v in the corpus. We require each negative n' to have a similar frequency to its corresponding n. This prevents our learning algorithm from focusing on any accidental frequency-based bias. We mix in K negatives for each positive, sampling without replacement to create all the negatives for a particular predicate. For each v, $\frac{1}{K+1}$ of its examples will be positive. The threshold τ represents a trade-off between capturing a large number of positive pairs and ensuring these pairs have good association. Similarly, K is a trade-off between the number of examples and the computational efficiency. Ultimately, these parameters should be optimized for task performance.

Of course, some negatives will actually be plausible arguments that were unobserved due to sparseness. Fortunately, modern discriminative methods like soft-margin SVMs can learn in the face of label error by allowing slack, subject to a tunable regularization penalty (Cortes and Vapnik, 1995).

If MI is a sparse and imperfect model of SP, what can DSP gain by training on MI's scores? We can regard DSP as learning a view of SP that is orthogonal to MI, in a co-training sense (Blum and Mitchell, 1998). MI labels the data based solely on co-occurrence; DSP uses these labels to identify other regularities – ones that extend beyond co-occurring words. For example, many instances of n where $\mathrm{MI}(\textit{eat},n)>\tau$ also have $\mathrm{MI}(\textit{buy},n)>\tau$ and $\mathrm{MI}(\textit{cook},n)>\tau$. Also, compared to other nouns, a disproportionate number of *eat*-nouns are lower-case, single-token words, and they rarely contain digits, hyphens, or begin with a human first name like *Bob*. DSP encodes these interdependent properties as features in a linear classifier. This classifier can score any noun as a plausible argument of *eat* if indicative features are present; MI can only assign high plausibility to observed (*eat*,n) pairs. Similarity-smoothed models can make use of the regularities across similar verbs, but not the finer-grained string- and token-based features.

Our training examples are similar to the data created for *pseudodisambiguation*, the usual evaluation task for SP models (Erk, 2007; Keller and Lapata, 2003; Rooth et al., 1999). This data consists of triples (v,n,n') where v,n is a predicate-argument pair observed in the corpus and v,n' has not been observed. The models score correctly if they rank observed (and thus plausible) arguments above corresponding unobserved (and thus likely implausible) ones. We refer to this as *Pairwise Disambiguation*. Unlike this task, we classify each predicate-argument pair independently as plausible/implausible. We also use MI rather than frequency to define the positive pairs, ensuring that the positive pairs truly have a statistical association, and are not simply the result of parser error or noise.[1]

3.2 Partitioning for Efficient Training

After creating our positive and negative training pairs, we must select a feature representation for our examples. Let $\mathbf{\Phi}$ be a mapping from a predicate-argument pair (v,n) to a feature vector, $\mathbf{\Phi} : (v,n) \rightarrow \langle\phi_1 ... \phi_k\rangle$. Predictions are made based on a weighted combination of the features, $y = \boldsymbol{\lambda}\cdot\mathbf{\Phi}(v,n)$, where $\boldsymbol{\lambda}$ is our learned weight vector.

We can make training significantly more efficient by using a special form of attribute-value features. Let every feature ϕ_i be of the form $\phi_i(v,n) = \langle v = \hat{v} \wedge f(n)\rangle$. That is, every feature is an intersection of the occurrence of a particular predicate, $\hat{v}$, and some feature of the argument $f(n)$. For example, a feature for a verb-object pair might be, "the verb is *eat* and the object is lower-case." In this representation, features for one predicate will be completely independent from those for every other predicate. Thus rather than a single training procedure, we can actually partition the examples by predicate, and train a

[1]For a fixed verb, MI is proportional to Keller and Lapata (2003)'s conditional probability scores for pseudodisambiguation of (v,n,n') triples: $\Pr(v|n) = \Pr(v,n)/\Pr(n)$, which was shown to be a better measure of association than co-occurrence frequency $f(v,n)$. Normalizing by $\Pr(v)$ (yielding MI) allows us to use a constant threshold across all verbs. MI was also recently used for inference-rule SPs by Pantel et al. (2007).

classifier for each predicate independently. The prediction becomes $y^v = \boldsymbol{\lambda}^v \cdot \boldsymbol{\Phi}^v(n)$, where $\boldsymbol{\lambda}^v$ are the learned weights corresponding to predicate v and all features $\boldsymbol{\Phi}^v(n)$=$\mathbf{f}(n)$ depend on the argument only.

Some predicate partitions may have insufficient examples for training. Also, a predicate may occur in test data that was unseen during training. To handle these instances, we decided to cluster low-frequency predicates. In our experiments assigning SP to verb-object pairs, we cluster all verbs that have less than 250 positive examples, using clusters generated by the CBC algorithm (Pantel and Lin, 2002). For example, the low-frequency verbs *incarcerate*, *parole*, and *court-martial* are all mapped to the same partition, while more-frequent verbs like *arrest* and *execute* each have their own partition. About 5.5% of examples are clustered, corresponding to 30% of the 7367 total verbs. 40% of verbs (but only 0.6% of examples) were not in any CBC cluster; these were mapped to a single backoff partition.

The parameters for each partition, $\boldsymbol{\lambda}^v$, can be trained with any supervised learning technique. We use SVM (Section 4.1) because it is effective in similar high-dimensional, sparse-vector settings, and has an efficient implementation (Joachims, 1999). In SVM, the sign of y^v gives the classification. We can also use the scalar y^v as our DSP score (i.e. the positive distance from the separating SVM hyperplane).

3.3 Features

This section details our argument features, $\mathbf{f}(n)$, for assigning verb-object selectional preference. For a verb predicate (or partition) v and object argument n, the form of our classifier is $y^v = \sum_i \lambda_i^v f_i(n)$.

3.3.1 Verb co-occurrence

We provide features for the empirical probability of the noun occurring as the object argument of other verbs, Pr$(n|v')$. If we were to only use these features (indexing the feature weights by each verb v'), the form of our classifier would be:

$$y^v = \sum_{v'} \lambda_{v'}^v \mathrm{Pr}(n|v') \qquad (3)$$

Note the similarity between Equation (3) and Equation (1). Now the feature weights, $\lambda_{v'}^v$, take the role of the similarity function, *Sim*(v', v). Unlike Equation (1), however, these weights are not set by an external similarity algorithm, but are optimized to discriminate the positive and negative training examples. We need not restrict ourselves to a short list of similar verbs; we include $\mathrm{Pr}_{obj}(n|v')$ features for every verb that occurs more than 10 times in our corpus. $\lambda_{v'}^v$ may be positive or negative, depending on the relation between v' and v. We also include features for the probability of the noun occurring as the *subject* of other verbs, $\mathrm{Pr}_{subj}(n|v')$. For example, nouns that can be the object of *eat* will also occur as the subject of *taste* and *contain*. Other contexts, such as adjectival and nominal predicates, could also aid the prediction, but have not yet been investigated.

The advantage of tuning similarity to the application of interest has been shown previously by Weeds and Weir (2005). They optimize a few meta-parameters separately for the tasks of thesaurus generation and pseudodisambiguation. Our approach, on the other hand, discriminatively sets millions of individual similarity values. Like Weeds and Weir (2005), our similarity values are asymmetric.

3.3.2 String-based

We include several simple character-based features of the noun string: the number of tokens, the case, and whether it contains digits, hyphens, an apostrophe, or other punctuation. We also include a feature for the first and last token, and fire indicator features if any token in the noun occurs on in-house lists of given names, family names, cities, provinces, countries, corporations, languages, etc. We also fire a feature if a token is a corporate designation (like *inc.* or *ltd.*) or a human one (like *Mr.* or *Sheik*).

3.3.3 Semantic classes

Motivated by previous SP models that make use of semantic classes, we generated word clusters using CBC (Pantel and Lin, 2002) on a 10 GB corpus, giving 3620 clusters. If a noun belongs in a cluster, a corresponding feature fires. If a noun is in none of the clusters, a *no-class* feature fires.

As an example, CBC cluster 1891 contains:

> sidewalk, driveway, roadway, footpath, bridge, highway, road, runway, street, alley, path, Interstate, . . .

In our training data, we have examples like *widen highway*, *widen road* and *widen motorway*. If we see that we can widen a highway, we learn that we can also widen a sidewalk, bridge, runway, etc.

We also made use of the person-name/instance pairs automatically extracted by Fleischman et al. (2003).[2] This data provides counts for pairs such as "Edwin Moses, *hurdler*" and "William Farley, *industrialist*." We have features for all *concepts* and therefore learn their association with each verb.

4 Experiments and Results

4.1 Set up

We parsed the 3 GB AQUAINT corpus (Voorhees, 2002) using Minipar (Lin, 1998b), and collected verb-object and verb-subject frequencies, building an empirical MI model from this data. Verbs and nouns were converted to their (possibly multi-token) *root*, and string case was preserved. Passive subjects (*the car was bought*) were converted to objects (*bought car*). We set the MI-threshold, τ, to be 0, and the negative-to-positive ratio, K, to be 2.

Numerous previous pseudodisambiguation evaluations only include arguments that occur between 30 and 3000 times (Erk, 2007; Keller and Lapata, 2003; Rooth et al., 1999). Presumably the lower bound is to help ensure the negative argument is unobserved because it is unsuitable, not because of data sparseness. We wish to use our model on arguments of any frequency, including those that never occurred in the training corpus (and therefore have empty co-occurrence features (Section 3.3.1)). We proceed as follows: first, we exclude pairs whenever the noun occurs less than 3 times in our corpus, removing many misspellings and other noun noise. Next, we omit verb co-occurrence features for nouns that occur less than 10 times, and instead fire a low-count feature. When we move to a new corpus, previously-unseen nouns are treated like these low-count training nouns.

This processing results in a set of 6.8 million pairs, divided into 2318 partitions (192 of which are verb clusters (Section 3.2)). For each partition, we take 95% of the examples for training, 2.5% for development and 2.5% for a final unseen test set. We provide full results for two models: DSP_{cooc} which only uses the verb co-occurrence features, and DSP_{all} which uses all the features mentioned in Section 3.3. Feature values are normalized within each feature type. We train our (linear kernel) discriminative models using SVM^{light} (Joachims, 1999) on each partition, but set meta-parameters C (regularization) and j (cost of positive vs. negative misclassifications: max at $j{=}2$) on the macro-averaged score across all development partitions. Note that we can not use the development set to optimize τ and K because the development examples are obtained *after* setting these values.

[2]Available at http://www.mit.edu/˜mbf/instances.txt.gz

4.2 Feature weights

It is interesting to inspect the feature weights returned by our system. In particular, the weights on the verb co-occurrence features (Section 3.3.1) provide a high-quality, argument-specific similarity-ranking of other verb contexts. The DSP parameters for *eat*, for example, place high weight on features like $\Pr(n|\textit{braise})$, $\Pr(n|\textit{ration})$, and $\Pr(n|\textit{garnish})$. Lin (1998a)'s similar word list for *eat* misses these but includes *sleep* (ranked 6) and *sit* (ranked 14), because these have similar *subjects* to *eat*. Discriminative, context-specific training seems to yield a better set of similar predicates, e.g. the highest-ranked contexts for DSP_{cooc} on the verb *join*,[3]

> lead 1.42, rejoin 1.39, form 1.34, belong to 1.31, found 1.31, quit 1.29, guide 1.19, induct 1.19, launch (*subj*) 1.18, work at 1.14

give a better $\text{SIMS}(\textit{join})$ for Equation (1) than the top similarities returned by (Lin, 1998a):

> participate 0.164, lead 0.150, return to 0.148, say 0.143, rejoin 0.142, sign 0.142, meet 0.142, include 0.141, leave 0.140, work 0.137

Other features are also weighted intuitively. Note that case is a strong indicator for some arguments, for example the weight on being lower-case is high for *become* (0.972) and *eat* (0.505), but highly negative for *accuse* (-0.675) and *embroil* (-0.573) which often take names of people and organizations.

4.3 Pseudodisambiguation

We first evaluate DSP on disambiguating positives from pseudo-negatives, comparing to recently-

[3]Which all correspond to nouns occurring in the object position of the verb (e.g. $\Pr_{obj}(n|\textit{lead})$), except "launch (*subj*)" which corresponds to $\Pr_{subj}(n|\textit{launch})$.

System	*MacroAvg*			*MicroAvg*			*Pairwise*	
	P	R	F	P	R	F	Acc	Cov
Dagan et al. (1999)	0.36	**0.90**	0.51	0.68	**0.92**	0.78	0.58	0.98
Erk (2007)	0.49	0.66	0.56	0.70	0.82	0.76	0.72	0.83
Keller and Lapata (2003)	**0.72**	0.34	0.46	**0.80**	0.50	0.62	0.80	0.57
DSP_{cooc}	0.53	0.72	0.61	0.73	0.94	**0.82**	0.77	**1.00**
DSP_{all}	0.60	0.71	**0.65**	0.77	0.90	**0.83**	**0.81**	**1.00**

Table 1: Pseudodisambiguation results averaged across each example (*MacroAvg*), weighted by word frequency (*MicroAvg*), plus coverage and accuracy of pairwise competition (*Pairwise*).

proposed systems that also require no manually-compiled resources like WordNet. We convert Dagan et al. (1999)'s similarity-smoothed probability to MI by replacing the empirical $\text{Pr}(n|v)$ in Equation (2) with the smoothed Pr_{SIM} from Equation (1). We also test an MI model inspired by Erk (2007):

$$\text{MI}_{\text{SIM}}(n, v) = \log \sum_{n' \in \text{SIMS}(n)} \mathit{Sim}(n', n) \frac{\text{Pr}(v, n')}{\text{Pr}(v)\text{Pr}(n')}$$

We gather similar words using Lin (1998a), mining similar verbs from a comparable-sized parsed corpus, and collecting similar nouns from a broader 10 GB corpus of English text.[4]

We also use Keller and Lapata (2003)'s approach to obtaining web-counts. Rather than mining parse trees, this technique retrieves counts for the pattern "V Det N" in raw online text, where V is any inflection of the verb, Det is *the*, *a*, or the empty string, and N is the singular or plural form of the noun. We compute a web-based MI by collecting $\text{Pr}(n, v)$, $\text{Pr}(n)$, and $\text{Pr}(v)$ using all inflections, except we only use the root form of the noun. Rather than using a search engine, we obtain counts from the Google Web 5-gram Corpus.[5]

All systems are thresholded at zero to make a classification. Unlike DSP, the comparison systems may not be able to provide a score for each example. The similarity-smoothed examples will be undefined if $\text{SIMS}(w)$ is empty. Also, the Keller and Lapata (2003) approach will be undefined if the pair is unobserved on the web. As a reasonable default for these cases, we assign them a negative decision.

We evaluate disambiguation using precision (P), recall (R), and their harmonic mean, F-Score (F). Table 1 gives the results of our comparison. In the *MacroAvg* results, we weight each example equally. For *MicroAvg*, we weight each example by the frequency of the noun. To more directly compare with previous work, we also reproduced *Pairwise Disambiguation* by randomly pairing each positive with one of the negatives and then evaluating each system by the percentage it ranks correctly (Acc). For the comparison approaches, if one score is undefined, we choose the other one. If both are undefined, we abstain from a decision. Coverage (Cov) is the percent of pairs where a decision was made.[6]

Our simple system with only verb co-occurrence features, DSP_{cooc}, outperforms all comparison approaches. Using the richer feature set in DSP_{all} results in a statistically significant gain in performance, up to an F-Score of 0.65 and a pairwise disambiguation accuracy of 0.81.[7] DSP_{all} has both broader coverage and better accuracy than all competing approaches. In the remainder of the experiments, we use DSP_{all} and refer to it simply as DSP.

Some errors are because of plausible but unseen arguments being used as test-set pseudo-negatives. For example, for the verb *damage*, DSP's three most high-scoring false positives are the nouns *jetliner*, *carpet*, and *gear*. While none occur with *damage* in

[4]For both the similar-noun and similar-verb smoothing, we only smooth over similar pairs *that occurred in the corpus*. While averaging over all similar pairs tends to underestimate the probability, averaging over only the observed pairs tends to overestimate it. We tested both and adopt the latter because it resulted in better performance on our development set.

[5]Available from the LDC as LDC2006T13. This collection was generated from approximately 1 trillion tokens of online text. Unfortunately, tokens appearing less than 200 times have been mapped to the ⟨UNK⟩ symbol, and only N-grams appearing more than 40 times are included. Unlike results from search engines, however, experiments with this corpus are replicable.

[6]I.e. we use the "half coverage" condition from Erk (2007).

[7]The differences between DSP_{all} and all comparison systems are statistically significant (McNemar's test, $p<0.01$).

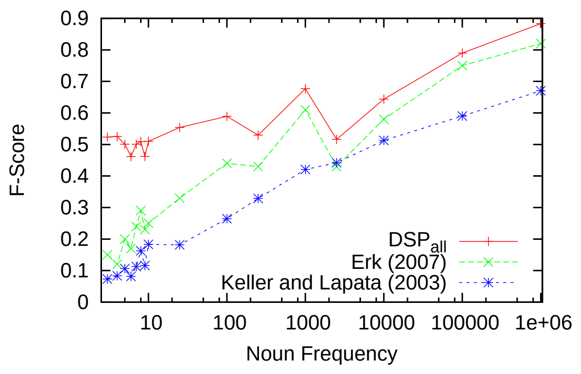


Figure 1: Disambiguation results by noun frequency.

our corpus, all intuitively satisfy the verb's SPs.

MacroAvg performance is worse than *MicroAvg* because all systems perform better on frequent nouns. When we plot F-Score by noun frequency (Figure 1), we see that DSP outperforms comparison approaches across all frequencies, but achieves its biggest gains on the low-frequency nouns. A richer feature set allows DSP to make correct inferences on examples that provide minimal co-occurrence data. These are also the examples for which we would expect co-occurrence models like MI to fail.

As a further experiment, we re-trained DSP but with only the string-based features removed. Overall macro-averaged F-score dropped from 0.65 to 0.64 (a statistically significant reduction in performance). The system scored nearly identically to DSP on the high-frequency nouns, but performed roughly 15% worse on the nouns that occurred less than ten times. This shows that the string-based features are important for selectional preference, and particularly helpful for low-frequency nouns.

4.4 Human Plausibility

Table 2 compares some of our systems on data used by Resnik (1996) (also Appendix 2 in Holmes et al. (1989)). The plausibility of these pairs was initially judged based on the experimenters' intuitions, and later confirmed in a human experiment. We include the scores of Resnik's system, and note that its errors were attributed to sense ambiguity and other limitations of class-based approaches (Resnik, 1996).[8]

[8]For example, *warn-engine* scores highly because engines are in the class *entity*, and physical entities (e.g. people) are often objects of *warn*. Unlike DSP, Resnik's approach cannot learn that for *warn*, "the property of being a person is more important than the property of being an entity" (Resnik, 1996).

Seen Criteria	Unseen Verb-Object Freq.				
	All	$= 1$	$= 2$	$= 3$	> 3
MI > 0	0.44	0.33	0.57	0.70	0.82
Freq. > 0	0.57	0.45	0.76	0.89	0.96
DSP > 0	0.73	0.69	0.80	0.85	0.88

Table 3: Recall on identification of Verb-Object pairs from an unseen corpus (divided by pair frequency).

The other comparison approaches also make a number of mistakes, which can often be traced to a misguided choice of similar word to smooth with.

We also compare to our empirical MI model, trained on our parsed corpus. Although Resnik (1996) reported that 10 of the 16 plausible pairs did not occur in his training corpus, all of them occurred in ours and hence MI gives very reasonable scores on the plausible objects. It has no statistics, however, for many of the implausible ones. DSP can make finer decisions than MI, recognizing that "warning an engine" is more absurd than "judging a climate."

4.5 Unseen Verb-Object Identification

We next compare MI and DSP on a much larger set of plausible examples, and also test how well the models generalize across data sets. We took the MI and DSP systems trained on AQUAINT and asked them to rate observed (and thus likely plausible) verb-object pairs taken from an unseen corpus. We extracted the pairs by parsing the San Jose Mercury News (SJM) section of the TIPSTER corpus (Harman, 1992). Each unique verb-object pair is a single instance in this evaluation.

Table 3 gives recall across all pairs (**All**) and grouped by pair-frequency in the unseen corpus (1, 2, 3, >3). DSP accepts far more pairs than MI (73% vs. 44%), even far more than a system that accepts any previously observed verb-object combination as plausible (57%). Recall is higher on more frequent verb-object pairs, but 70% of the pairs occurred only once in the corpus. Even if we smooth MI by smoothing $\Pr(n|v)$ in Equation 2 using modified KN-smoothing (Chen and Goodman, 1998), the recall of MI>0 on SJM only increases from 44.1% to 44.9%, still far below DSP. Frequency-based models have fundamentally low coverage. As fur-

Verb	Plaus./Implaus.	Resnik	Dagan et al.	Erk	MI	DSP
see	friend/method	5.79/-0.01	*0.20/1.40**	0.46/-0.07	1.11/-0.57	0.98/0.02
read	article/fashion	6.80/-0.20	3.00/0.11	3.80/1.90	4.00/—	2.12/-0.65
find	label/fever	1.10/0.22	*1.50/2.20**	0.59/0.01	0.42/0.07	1.61/0.81
hear	story/issue	*1.89/1.89**	*0.66/1.50**	*2.00/2.60**	2.99/-1.03	1.66/0.67
write	letter/market	7.26/0.00	2.50/-0.43	3.60/-0.24	5.06/-4.12	3.08/-1.31
urge	daughter/contrast	*1.14/1.86**	*0.14/1.60**	*1.10/3.60**	-0.95/—	-0.34/-0.62
warn	driver/engine	4.73/3.61	1.20/0.05	2.30/0.62	2.87/—	2.00/-0.99
judge	contest/climate	1.30/0.28	*1.50/1.90**	*1.70/1.70**	3.90/—	1.00/0.51
teach	language/distance	*1.87/1.86*	2.50/1.30	3.60/2.70	3.53/—	1.86/0.19
show	sample/travel	1.44/0.41	1.60/0.14	0.40/-0.82	0.53/-0.49	1.00/-0.83
expect	visit/mouth	*0.59/5.93**	*1.40/1.50**	1.40/0.37	1.05/-0.65	1.44/-0.15
answer	request/tragedy	4.49/3.88	2.70/1.50	3.10/-0.64	2.93/—	1.00/0.01
recognize	author/pocket	*0.50/0.50**	*0.03/0.37**	*0.77/1.30**	0.48/—	1.00/0.00
repeat	comment/journal	*1.23/1.23**	2.30/1.40	2.90/—	2.59/—	1.00/-0.48
understand	concept/session	*1.52/1.51*	2.70/0.25	2.00/-0.28	3.96/—	2.23/-0.46
remember	reply/smoke	1.31/0.20	2.10/1.20	*0.54/2.60**	1.13/-0.06	1.00/-0.42

Table 2: Selectional ratings for plausible/implausible direct objects (Holmes et al., 1989). Mistakes are marked with an asterisk (*), undefined scores are marked with a dash (—). Only DSP is completely defined and completely correct.

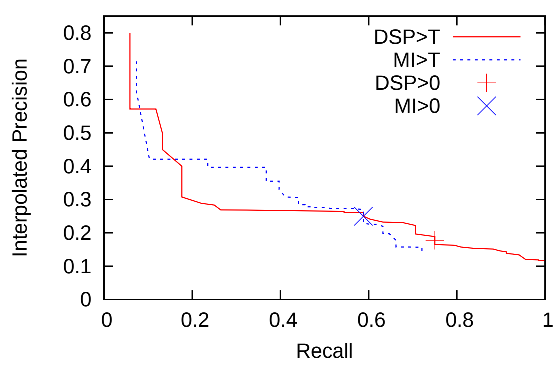


Figure 2: Pronoun resolution precision-recall on MUC.

ther evidence, if we build a model of MI on the SJM corpus and use it in our pseudodisambiguation experiment (Section 4.3), MI>0 gets a *MacroAvg* precision of 86% but a *MacroAvg* recall of only 12%.[9]

4.6 Pronoun Resolution

Finally, we evaluate DSP on a common application of selectional preferences: choosing the correct antecedent for pronouns in text (Dagan and Itai, 1990; Kehler et al., 2004). We study the cases where a pronoun is the direct object of a verb predicate, v. A pronoun's antecedent must obey v's selectional preferences. If we have a better model of SP, we should be able to better select pronoun antecedents.

We parsed the MUC-7 (1997) coreference corpus and extracted all pronouns in a direct object relation. For each pronoun, p, modified by a verb, v, we extracted all preceding nouns within the current or previous sentence. Thirty-nine anaphoric pronouns had an antecedent in this window and are used in the evaluation. For each p, let $N(p)^+$ by the set of preceding nouns coreferent with p, and let $N(p)^-$ be the remaining non-coreferent nouns. We take all (v, n^+) where $n^+ \in N(p)^+$ as positive, and all other pairs (v, n^-), $n^- \in N(p)^-$ as negative.

We compare MI and DSP on this set, classifying every (v, n) with MI>T (or DSP>T) as positive. By varying T, we get a precision-recall curve (Figure 2). Precision is low because, of course, there are many nouns that satisfy the predicate's SPs that are not coreferent. DSP>0 has both a higher recall and higher precision than accepting every pair previously seen in text (the right-most point on MI>T). The DSP>T system achieves higher precision than MI>T for points where recall is greater than 60% (where MI<0). Interestingly, the recall of MI>0 is

[9]Recall that even the Keller and Lapata (2003) system, built on the world's largest corpus, achieves only 34% recall (Table 1) (with only 48% of positives and 27% of all pairs previously observed, but see Footnote 5).

System	Acc
Most-Recent Noun	17.9%
Maximum MI	28.2%
Maximum DSP	38.5%

Table 4: Pronoun resolution accuracy on nouns in current or previous sentence in MUC.

higher here than it is for general verb-objects (Section 4.5). On the subset of pairs with strong empirical association (MI$>$0), MI generally outperforms DSP at equivalent recall values.

We next compare MI and DSP as stand-alone pronoun resolution systems (Table 4). As a standard baseline, for each pronoun, we choose the most recent noun in text as the pronoun's antecedent, achieving 17.9% resolution accuracy. This baseline is quite low because many of the most-recent nouns are subjects of the pronoun's verb phrase, and therefore resolution violates syntactic coreference constraints. If instead we choose the previous noun with the highest MI as antecedent, we get an accuracy of 28.2%, while choosing the previous noun with the highest DSP achieves 38.5%. DSP resolves 37% more pronouns correctly than MI. We leave as future work a full-scale pronoun resolution system that incorporates both MI and DSP as backed-off, interpolated, or separate semantic features.

5 Conclusions and Future Work

We have presented a simple, effective model of selectional preference based on discriminative training. Supervised techniques typically achieve higher performance than unsupervised models, and we duplicate these gains with DSP. Here, however, these gains come at no additional labeling cost, as training examples are generated automatically from unlabeled text. DSP allows an arbitrary combination of features, including verb co-occurrence features that yield high-quality similar-word lists as latent output. This work only scratches the surface of possible feature mining; information from WordNet relations, Wikipedia categories, or parallel corpora could also provide valuable clues to SP. Also, if any other system were to exceed DSP's performance, it could also be included as one of DSP's features.

It would be interesting to expand our co-occurrence features, including co-occurrence counts across more grammatical relations and using counts from external, unparsed corpora like the world wide web. We could also reverse the role of noun and verb in our training, having verb-specific features and discriminating separately for each argument noun. The latent information would then be lists of similar nouns.

Finally, note that while we focused on word-word co-occurrences, sense-sense SPs can also be learned with our algorithm. If our training corpus was sense-labeled, we could run our algorithm over the senses rather than the words. The resulting model would then require sense-tagged input if it were to be used within an application like parsing or coreference resolution. Also, like other models of SP, our technique can also be used for sense disambiguations: the weightings on our semantic class features indicate, for a particular noun, which of its senses (classes) is most compatible with each verb.

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

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Daniel M. Bikel. 2004. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4):479–511.

John Blitzer, Amir Globerson, and Fernando Pereira. 2005. Distributed latent variable models of lexical co-occurrences. In *AISTATS*.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of COLT*, pages 92–100.

Carsten Brockmann and Mirella Lapata. 2003. Evaluating and combining approaches to selectional preference acquisition. In *EACL*, pages 27–34.

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. TR-10-98, Harvard University.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Stephen Clark and David Weir. 2002. Class-based probability estimation using a semantic hierarchy. *Computational Linguistics*, 28(2):187–206.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297.

Ido Dagan and Alan Itai. 1990. Automatic processing of large corpora for the resolution of anaphora references. In *COLING*, volume 3, pages 330–332.

Ido Dagan, Lillian Lee, and Fernando C. N. Pereira. 1999. Similarity-based models of word cooccurrence probabilities. *Machine Learning*, 34(1-3):43–69.

Katrin Erk. 2007. A simple, similarity-based model for selectional preference. In *ACL*, pages 216–223.

Yair Even-Zohar and Dan Roth. 2000. A classification approach to word prediction. In *NAACL*, pages 124–131.

Michael Fleischman, Eduard Hovy, and Abdessamad Echihabi. 2003. Offline strategies for online question answering: answering questions before they are asked. In *ACL*, pages 1–7.

Donna Harman. 1992. The DARPA TIPSTER project. *ACM SIGIR Forum*, 26(2):26–28.

Virginia M. Holmes, Laurie Stowe, and Linda Cupples. 1989. Lexical expectations in parsing complement-verb sentences. *Journal of Memory and Language*, 28:668–689.

Thorsten Joachims. 1999. Making large-scale Support Vector Machine learning practical. In B. Schölkopf and C. Burges, editors, *Advances in Kernel Methods: Support Vector Machines*, pages 169–184. MIT-Press.

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

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

Hang Li and Naoki Abe. 1998. Generalizing case frames using a thesaurus and the MDL principle. *Computational Linguistics*, 24(2):217–244.

Dekang Lin. 1998a. Automatic retrieval and clustering of similar words. In *COLING-ACL*, pages 768–773.

Dekang Lin. 1998b. Dependency-based evaluation of MINIPAR. In *LREC Workshop on the Evaluation of Parsing Systems*.

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

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

Daisuke Okanohara and Jun'ichi Tsujii. 2007. A discriminative language model with pseudo-negative samples. In *ACL*, pages 73–80.

Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In *KDD*, pages 613–619.

Patrick Pantel, Rahul Bhagat, Bonaventura Coppola, Timothy Chklovski, and Eduard Hovy. 2007. ISP: Learning inferential selectional preferences. In *NAACL-HLT*, pages 564–571.

Philip Resnik. 1996. Selectional constraints: An information-theoretic model and its computational realization. *Cognition*, 61:127–159.

Basili Roberto, Diego De Cao, Paolo Marocco, and Marco Pennacchiotti. 2007. Learning selectional preferences for entailment or paraphrasing rules. In *RANLP*.

Mats Rooth, Stefan Riezler, Detlef Prescher, Glenn Carroll, and Franz Beil. 1999. Inducing a semantically annotated lexicon via EM-based clustering. In *ACL*, pages 104–111.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: training log-linear models on unlabeled data. In *ACL*, pages 354–362.

Ellen Voorhees. 2002. Overview of the TREC 2002 question answering track. In *Proceedings of the Eleventh Text REtrieval Conference (TREC)*.

Qin Iris Wang, Dale Schuurmans, and Dekang Lin. 2005. Strictly lexical dependency parsing. In *International Workshop on Parsing Technologies*, pages 152–159.

Julie Weeds and David Weir. 2005. Co-occurrence retrieval: a flexible framework for lexical distributional similarity. *Computational Linguistics*, 31(4):439–475.

Dependency-based Semantic Role Labeling of PropBank

Richard Johansson and **Pierre Nugues**
Lund University, Sweden
{richard, pierre}@cs.lth.se

Abstract

We present a PropBank semantic role labeling system for English that is integrated with a dependency parser. To tackle the problem of joint syntactic–semantic analysis, the system relies on a syntactic and a semantic subcomponent. The syntactic model is a projective parser using pseudo-projective transformations, and the semantic model uses global inference mechanisms on top of a pipeline of classifiers. The complete syntactic–semantic output is selected from a candidate pool generated by the subsystems.

We evaluate the system on the CoNLL-2005 test sets using segment-based and dependency-based metrics. Using the segment-based CoNLL-2005 metric, our system achieves a near state-of-the-art F1 figure of 77.97 on the WSJ+Brown test set, or 78.84 if punctuation is treated consistently. Using a dependency-based metric, the F1 figure of our system is 84.29 on the test set from CoNLL-2008. Our system is the first dependency-based semantic role labeler for PropBank that rivals constituent-based systems in terms of performance.

1 Introduction

Automatic semantic role labeling (SRL), the task of determining *who* does *what* to *whom*, is a useful intermediate step in NLP applications performing semantic analysis. It has obvious applications for template-filling tasks such as information extraction and question answering (Surdeanu et al., 2003; Moschitti et al., 2003). It has also been used in prototypes of NLP systems that carry out complex reasoning, such as entailment recognition systems (Haghighi et al., 2005; Hickl et al., 2006). In addition, role-semantic features have recently been used to extend vector-space representations in automatic document categorization (Persson et al., 2008).

The NLP community has recently devoted much attention to developing accurate and robust methods for performing role-semantic analysis automatically, and a number of multi-system evaluations have been carried out (Litkowski, 2004; Carreras and Màrquez, 2005; Baker et al., 2007; Surdeanu et al., 2008). Following the seminal work of Gildea and Jurafsky (2002), there have been many extensions in machine learning models, feature engineering (Xue and Palmer, 2004), and inference procedures (Toutanova et al., 2005; Surdeanu et al., 2007; Punyakanok et al., 2008).

With very few exceptions (e.g. Collobert and Weston, 2007), published SRL methods have used some sort of syntactic structure as input (Gildea and Palmer, 2002; Punyakanok et al., 2008). Most systems for automatic role-semantic analysis have used constituent syntax as in the Penn Treebank (Marcus et al., 1993), although there has also been much research on the use of shallow syntax (Carreras and Màrquez, 2004) in SRL.

In comparison, dependency syntax has received relatively little attention for the SRL task, despite the fact that dependency structures offer a more transparent encoding of predicate–argument relations. Furthermore, the few systems based on dependencies that have been presented have generally performed much worse than their constituent-based counterparts. For instance, Pradhan et al. (2005) reported that a system using a rule-based dependency parser achieved much inferior results compared to a system using a state-of-the-art statistical constituent parser: The F-measure on WSJ section 23 dropped from 78.8 to 47.2, or from 83.7 to 61.7 when using a head-based evaluation. In a similar vein, Swanson and Gordon (2006) reported that parse tree path features extracted from a rule-based dependency parser are much less reliable than those from a modern constituent parser.

In contrast, we recently carried out a detailed comparison (Johansson and Nugues, 2008b) between constituent-based and dependency-based SRL systems for FrameNet, in which the results of the two types of systems where almost equivalent when using modern statistical dependency parsers. We suggested that the previous lack of progress in dependency-based SRL was due to low parsing accuracy. The experiments showed that the grammatical function information available in dependency representations results in a steeper learning curve when training semantic role classifiers, and it also seemed that the dependency-based role classifiers were more resilient to lexical problems caused by change of domain.

The recent CoNLL-2008 Shared Task (Surdeanu et al., 2008) was an attempt to show that SRL can be accurately carried out using only dependency syntax. However, these results are not easy to compare to previously published results since the task definitions and evaluation metrics were different.

This paper compares the best-performing system in the CoNLL-2008 Shared Task (Johansson and Nugues, 2008a) with previously published constituent-based SRL systems. The system carries out joint dependency-syntactic and semantic analysis. We first describe its implementation in Section 2, and then compare the system with the best system in the CoNLL-2005 Shared Task in Section 3. Since the outputs of the two systems are different, we carry out two types of evaluations: first by using the traditional segment-based metric used in the CoNLL-2005 Shared Task, and then by using the dependency-based metric from the CoNLL-2008 Shared Task. Both evaluations require a transformation of the output of one system: For the segment-based metric, we have to convert the dependency-based output to segments; and for the dependency-based metric, a head-finding procedure is needed to select heads in segments. For the first time for a system using only dependency syntax, we report results for PropBank-based semantic role labeling of English that are close to the state of the art, and for some measures even superior.

2 Syntactic–Semantic Dependency Analysis

The training corpus that we used is the dependency-annotated Penn Treebank from the 2008 CoNLL Shared Task on joint syntactic–semantic analysis (Surdeanu et al., 2008). Figure 1 shows a sentence annotated in this framework. The CoNLL task involved semantic analysis of predicates from PropBank (for verbs, such as *plan*) and NomBank (for nouns, such as *investment*); in this paper, we report the performance on PropBank predicates only since we compare our system with previously published PropBank-based SRL systems.

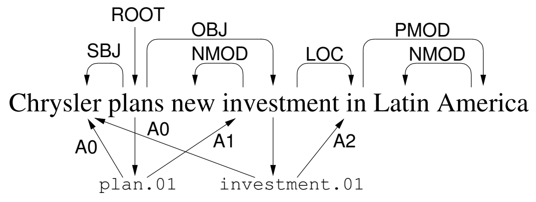


Figure 1: An example sentence annotated with syntactic and semantic dependency structures.

We model the problem of constructing a syntactic and a semantic graph as a task to be solved jointly. Intuitively, syntax and semantics are highly interdependent and semantic interpretation should help syntactic disambiguation, and joint syntactic–semantic analysis has a long tradition in deep-linguistic formalisms. Using a discriminative model, we thus formulate the problem of finding a syntactic tree $\hat{y}_{syn}$ and a semantic graph $\hat{y}_{sem}$ for a sentence $\boldsymbol{x}$ as maximizing a function F_{joint} that scores the complete syntactic–semantic structure:

$$\langle \hat{y}_{syn}, \hat{y}_{sem} \rangle = \arg \max_{y_{syn}, y_{sem}} F_{joint}(\boldsymbol{x}, y_{syn}, y_{sem})$$

The dependencies in the feature representation used to compute F_{joint} determine the tractability of the

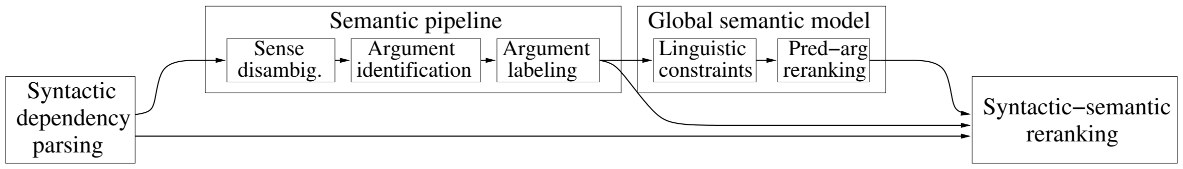


Figure 2: The architecture of the syntactic–semantic analyzer.

search procedure needed to perform the maximization. To be able to use complex syntactic features such as paths when predicting semantic structures, exact search is clearly intractable. This is true even with simpler feature representations – the problem is a special case of multi-headed dependency analysis, which is NP-hard even if the number of heads is bounded (Chickering et al., 1994).

This means that we must resort to a simplification such as an incremental method or a reranking approach. We chose the latter option and thus created syntactic and semantic submodels. The joint syntactic–semantic prediction is selected from a small list of candidates generated by the respective subsystems. Figure 2 shows the architecture.

2.1 Syntactic Submodel

We model the process of syntactic parsing of a sentence $\boldsymbol{x}$ as finding the parse tree $\hat{y}_{syn} = \arg\max_{y_{syn}} F_{syn}(\boldsymbol{x}, y_{syn})$ that maximizes a scoring function F_{syn}. The learning problem consists of fitting this function so that the cost of the predictions is as low as possible according to a cost function ρ_{syn}. In this work, we consider linear scoring functions of the following form:

$$F_{syn}(\boldsymbol{x}, y_{syn}) = \boldsymbol{\Psi}_{syn}(\boldsymbol{x}, y_{syn}) \cdot \boldsymbol{w}$$

where $\boldsymbol{\Psi}_{syn}(\boldsymbol{x}, y_{syn})$ is a numeric feature representation of the pair $(\boldsymbol{x}, y_{syn})$ and $\boldsymbol{w}$ a vector of feature weights. We defined the syntactic cost ρ_{syn} as the sum of link costs, where the link cost was 0 for a correct dependency link with a correct label, 0.5 for a correct link with an incorrect label, and 1 for an incorrect link.

A widely used discriminative framework for fitting the weight vector is the max-margin model (Taskar et al., 2003), which is a generalization of the well-known support vector machines to general cost-based prediction problems. Since the large number of training examples and features in our case make an exact solution of the max-margin optimization problem impractical, we used the online passive–aggressive algorithm (Crammer et al., 2006), which approximates the optimization process in two ways:

- The weight vector $\boldsymbol{w}$ is updated incrementally, one example at a time.
- For each example, only the most violated constraint is considered.

The algorithm is a margin-based variant of the perceptron (preliminary experiments show that it outperforms the ordinary perceptron on this task). Algorithm 1 shows pseudocode for the algorithm.

Algorithm 1 The Online PA Algorithm

```
input Training set $\mathcal{T} = \{(\boldsymbol{x}_t, y_t)\}_{t=1}^{T}$
      Number of iterations $N$
      Regularization parameter $C$
Initialize $w$ to zeros
repeat $N$ times
  for $(\boldsymbol{x}_t, y_t)$ in $\mathcal{T}$
    let $\tilde{y}_t = \arg\max_y F(\boldsymbol{x}_t, y) + \rho(y_t, y)$
    let $\tau_t = \min\left(C, \frac{F(\boldsymbol{x}_t,\tilde{y}_t) - F(\boldsymbol{x}_t,y_t) + \rho(y_t,\tilde{y}_t)}{\|\boldsymbol{\Psi}(\boldsymbol{x},y_t) - \boldsymbol{\Psi}(\boldsymbol{x},\tilde{y}_t)\|^2}\right)$
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \tau_t(\boldsymbol{\Psi}(\boldsymbol{x}, y_t) - \boldsymbol{\Psi}(\boldsymbol{x}, \tilde{y}_t))$
return $\boldsymbol{w}_{\text{average}}$
```

We used a C value of 0.01, and the number of iterations was 6.

2.1.1 Features and Search

The feature function $\boldsymbol{\Psi}_{syn}$ is a factored representation, meaning that we compute the score of the complete parse tree by summing the scores of its parts, referred to as *factors*:

$$\boldsymbol{\Psi}(\boldsymbol{x}, y) \cdot \boldsymbol{w} = \sum_{f \in y} \psi(\boldsymbol{x}, f) \cdot \boldsymbol{w}$$

We used a second-order factorization (McDonald and Pereira, 2006; Carreras, 2007), meaning that the factors are subtrees consisting of four links: the governor–dependent link, its sibling link, and the leftmost and rightmost dependent links of the dependent.

This factorization allows us to express useful features, but also forces us to adopt the expensive search procedure by Carreras (2007), which extends Eisner's span-based dynamic programming algorithm (1996) to allow second-order feature dependencies. This algorithm has a time complexity of $O(n^4)$, where n is the number of words in the sentence. The search was constrained to disallow multiple root links.

To evaluate the $\arg\max$ in Algorithm 1 during training, we need to handle the cost function ρ_{syn} in addition to the factor scores. Since the cost function ρ_{syn} is based on the cost of single links, this can easily be integrated into the factor-based search.

2.1.2 Handling Nonprojective Links

Although only 0.4% of the links in the training set are nonprojective, 7.6% of the sentences contain at least one nonprojective link. Many of these links represent long-range dependencies – such as *wh*-movement – that are valuable for semantic processing. Nonprojectivity cannot be handled by span-based dynamic programming algorithms. For parsers that consider features of single links only, the Chu-Liu/Edmonds algorithm can be used instead. However, this algorithm cannot be generalized to the second-order setting – McDonald and Pereira (2006) proved that this problem is NP-hard, and described an approximate greedy search algorithm.

To simplify implementation, we instead opted for the pseudo-projective approach (Nivre and Nilsson, 2005), in which nonprojective links are lifted upwards in the tree to achieve projectivity, and special trace labels are used to enable recovery of the nonprojective links at parse time. The use of trace labels in the pseudo-projective transformation leads to a proliferation of edge label types: from 69 to 234 in the training set, many of which occur only once. Since the running time of our parser depends on the number of labels, we used only the 20 most frequent trace labels.

2.2 Semantic Submodel

Our semantic model consists of three parts:

- A SRL classifier pipeline that generates a list of candidate predicate–argument structures.
- A constraint system that filters the candidate list to enforce linguistic restrictions on the global configuration of arguments.
- A global reranker that assigns scores to predicate–argument structures in the filtered candidate list.

Rather than training the models on gold-standard syntactic input, we created an automatically parsed training set by 5-fold cross-validation. Training on automatic syntax makes the semantic classifiers more resilient to parsing errors, in particular adjunct labeling errors.

2.2.1 SRL Pipeline

The SRL pipeline consists of classifiers for predicate disambiguation, argument identification, and argument labeling. For the predicate disambiguation classifiers, we trained one subclassifier for each lemma. All classifiers in the pipeline were L2-regularized linear logistic regression classifiers, implemented using the efficient LIBLINEAR package (Lin et al., 2008). For multiclass problems, we used the one-vs-all binarization method, which makes it easy to prevent outputs not allowed by the PropBank frame.

Since our classifiers were logistic, their output values could be meaningfully interpreted as probabilities. This allowed us to combine the scores from subclassifiers into a score for the complete predicate–argument structure. To generate the candidate lists used by the global SRL models, we applied beam search based on these scores using a beam width of 4.

The argument identification classifier was preceded by a pruning step similar to the constituent-based pruning by Xue and Palmer (2004).

The features used by the classifiers are listed in Table 1, and are described in Appendix A. We selected the feature sets by greedy forward subset selection.

Feature	PredDis	ArgId	ArgLab
PredWord	•		
PredLemma	•		
PredParentWord/POS	•		
ChildDepSet	•	•	•
ChildWordSet	•		
ChildWordDepSet	•		
ChildPOSSet	•		
ChildPOSDepSet	•		
DepSubcat	•		
PredRelToParent	•		
PredParentWord/POS	•		
PredLemmaSense		•	•
Voice		•	•
Position		•	•
ArgWord/POS		•	•
LeftWord/POS			•
RightWord/POS		•	•
LeftSiblingWord/POS			•
PredPOS		•	•
RelPath		•	•
VerbChainHasSubj		•	•
ControllerHasObj		•	
PredRelToParent		•	•
Function			•

Table 1: Classifier features in predicate disambiguation (PredDis), argument identification (ArgId), and argument labeling (ArgLab).

2.2.2 Linguistically Motivated Global Constraints

The following three global constraints were used to filter the candidates generated by the pipeline.

Core Argument Consistency. Core argument labels must not appear more than once.

Discontinuity Consistency. If there is a label C-X, it must be preceded by a label X.

Reference Consistency. If there is a label R-X and the label is inside an attributive relative clause, it must be preceded by a label X.

2.2.3 Predicate–Argument Reranker

Toutanova et al. (2005) have showed that a global model that scores the complete predicate–argument structure can lead to substantial performance gains. We therefore created a global SRL classifier using the following global features in addition to the features from the pipeline:

Core Argument Label Sequence. The complete sequence of core argument labels. The sequence also includes the predicate and voice, for instance `A0`+*break.01*/Active+`A1`.

Missing Core Argument Labels. The set of core argument labels declared in the PropBank frame that are not present in the predicate–argument structure.

Similarly to the syntactic submodel, we trained the global SRL model using the online passive–aggressive algorithm. The cost function ρ was defined as the number of incorrect links in the predicate–argument structure. The number of iterations was 20 and the regularization parameter C was 0.01. Interestingly, we noted that the global SRL model outperformed the pipeline even when no global features were added. This shows that the global learning model can correct label bias problems introduced by the pipeline architecture.

2.3 Syntactic–Semantic Reranking

As described previously, we carried out reranking on the candidate set of complete syntactic–semantic structures. To do this, we used the top 16 trees from the syntactic module and applied a linear model:

$$F_{joint}(\boldsymbol{x}, y_{syn}, y_{sem}) = \boldsymbol{\Psi}_{joint}(\boldsymbol{x}, y_{syn}, y_{sem}) \cdot \boldsymbol{w}$$

Our baseline joint feature representation $\boldsymbol{\Psi}_{joint}$ contained only three features: the log probability of the syntactic tree and the log probability of the semantic structure according to the pipeline and the global model, respectively. This model was trained on the complete training set using cross-validation. The probabilities were obtained using the multinomial logistic function ("softmax").

We carried out an initial experiment with a more complex joint feature representation, but failed to improve over the baseline. Time prevented us from exploring this direction conclusively.

3 Comparisons with Previous Results

To compare our results with previously published results in SRL, we carried out an experiment comparing our system to the top system (Punyakanok et al., 2008) in the CoNLL-2005 Shared Task. However, comparison is nontrivial since the output of the CoNLL-2005 systems was a set of labeled segments, while the CoNLL-2008 systems (including ours) produced labeled semantic dependency links.

To have a fair comparison of our link-based system against previous segment-based systems, we carried out a two-way evaluation: In the first evaluation, the dependency-based output was converted to segments and evaluated using the segment scorer from CoNLL-2005, and in the second evaluation, we applied a head-finding procedure to the output of a segment-based system and scored the result using the link-based CoNLL-2008 scorer.

It can be discussed which of the two metrics is most correlated with application performance. The traditional metric used in the CoNLL-2005 task treats SRL as a *bracketing problem*, meaning that the entities scored by the evaluation procedure are labeled snippets of text; however, it is questionable whether this is the proper way to evaluate a task whose purpose is to find *semantic relations* between logical entities. We believe that the same criticisms that have been leveled at the PARSEVAL metric for constituent structures are equally valid for the bracket-based evaluation of SRL systems. The inappropriateness of the traditional metric has led to a number of alternative metrics (Litkowski, 2004; Baker et al., 2007; Surdeanu et al., 2008).

3.1 Segment-based Evaluation

To be able to score the output of a dependency-based SRL system using the segment scorer, a conversion step is needed. Algorithm 2 shows how a set of segments is constructed from an argument dependency node. For each argument node, the algorithm computes the yield Y of the argument node, i.e. the set of dependency nodes to include in the bracketing. This set is then partitioned into contiguous parts, from which the segments are computed. In most cases, the yield is just the subtree dominated by the argument node. However, if the argument dominates the predicate, then the branch containing the predicate is removed.

Table 2 shows the performance figures of our system on the WSJ and Brown corpora: precision, recall, F_1-measure, and complete proposition accuracy (PP). These figures are compared to the best-performing system in the CoNLL-2005 Shared Task (Punyakanok et al., 2008), referred to as Punyakanok in the table, and the best result currently published (Surdeanu et al., 2007), referred to as Surdeanu. To validate the sanity of the segment creation algorithm, the table also shows the result of applying segment creation to gold-standard syntactic–semantic trees. We see that the two conversion procedures involved (constituent-to-dependency conversion by the CoNLL-2008 Shared Task organizers, and our dependency-to-segment conversion) work satisfactorily although the process is not completely lossless.

During inspection of the output, we noted that many errors arise from inconsistent punctuation attachment in PropBank/Treebank. We therefore normalized the segments to exclude punctuation at the beginning or end of a segment. The results of this evaluation is shown in Table 3. This table does not include the Surdeanu system since we did not have

Algorithm 2 Segment creation from an argument dependency node.

```
input Predicate node $p$, argument node $a$
if $a$ does not dominate $p$
    $Y \leftarrow \{n : a \text{ dominates } n\}$
else
    $c \leftarrow$ the child of $a$ that dominates $p$
    $Y \leftarrow \{n : a \text{ dominates } n\} \setminus \{n : c \text{ dominates } n\}$
end if
$S \leftarrow$ partition of $Y$ into contiguous subsets
return $\{(\text{min-index } s, \text{max-index } s) : s \in S\}$
```

WSJ	P	R	F1	PP
Our system	82.22	**77.72**	79.90	**57.24**
Punyakanok	82.28	76.78	79.44	53.79
Surdeanu	**87.47**	74.67	**80.56**	51.66
Gold standard	97.38	96.77	97.08	93.20

Brown	P	R	F1	PP
Our system	68.79	61.87	65.15	32.34
Punyakanok	73.38	**62.93**	67.75	32.34
Surdeanu	**81.75**	61.32	**70.08**	**34.33**
Gold standard	97.22	96.55	96.89	92.79

WSJ+Brown	P	R	F1	PP
Our system	80.50	**75.59**	77.97	**53.94**
Punyakanok	81.18	74.92	77.92	50.95
Surdeanu	**86.78**	72.88	**79.22**	49.36
Gold standard	97.36	96.75	97.05	93.15

Table 2: Evaluation with unnormalized segments.

access to its output.

WSJ	P	R	F1	PP
Our system	**82.95**	**78.40**	**80.61**	**58.65**
Punyakanok	82.67	77.14	79.81	54.55
Gold standard	97.85	97.24	97.54	94.34

Brown	P	R	F1	PP
Our system	70.84	63.71	67.09	**36.94**
Punyakanok	**74.29**	63.71	**68.60**	34.08
Gold standard	97.46	96.78	97.12	93.41

WSJ+Brown	P	R	F1	PP
Our system	81.39	**76.44**	**78.84**	**55.77**
Punyakanok	**81.63**	75.34	78.36	51.84
Gold standard	97.80	97.18	97.48	94.22

Table 3: Evaluation with normalized segments.

The results on the WSJ test set clearly show that dependency-based SRL systems can rival constituent-based systems in terms of performance – it clearly outperforms the Punyakanok system, and has a higher recall and complete proposition accuracy than the Surdeanu system. We interpret the high recall as a result of the dependency syntactic representation, which makes the parse tree paths simpler and thus the arguments easier to find.

For the Brown test set, on the other hand, the dependency-based system suffers from a low precision compared to the constituent-based systems. Our error analysis indicates that the domain change caused problems with prepositional attachment for the dependency parser – it is well-known that prepositional attachment is a highly lexicalized problem, and thus sensitive to domain changes. We believe that the reason why the constituent-based systems are more robust in this respect is that they utilize a combination strategy, using inputs from two different full constituent parsers, a clause bracketer, and a chunker. However, caution is needed when drawing conclusions from results on the Brown test set, which is only 7,585 words, compared to the 59,100 words in the WSJ test set.

3.2 Dependency-based Evaluation

It has previously been noted (Pradhan et al., 2005) that a segment-based evaluation may be unfavorable to a dependency-based system, and that an evaluation that scores argument *heads* may be more indicative of its true performance. We thus carried out an evaluation using the evaluation script of the CoNLL-2008 Shared Task. In this evaluation method, an argument is counted as correctly identified if its head and label are correct. Note that this is not equivalent to the segment-based metric: In a perfectly identified segment, we may still pick out the wrong head, and if the head is correct, we may infer an incorrect segment. The evaluation script also scores predicate disambiguation performance; we did not include this score since the 2005 systems did not output predicate sense identifiers.

Since CoNLL-2005-style segments have no internal tree structure, it is nontrivial to extract a head. It is conceivable that the output of the parsers used by the Punyakanok system could be used to extract heads, but this is not recommendable because the Punyakanok system is an ensemble system and a segment does not always exactly match a constituent in a parse tree. Furthermore, the CoNLL-2008 constituent-to-dependency conversion method uses a richer structure than just the raw constituents: empty categories, grammatical functions, and named entities. To recreate this additional information, we would have to apply automatic systems and end up with unreliable results.

Instead, we thus chose to find an *upper bound* on the performance of the segment-based system. We applied a simple head-finding procedure (Algorithm 3) to find a set of head nodes for each segment. Since the CoNLL-2005 output does not include dependency information, the algorithm uses gold-standard dependencies and intersects segments with the gold-standard segments. This will give us an upper bound, since if the segment contains the correct head, it will always be counted as correct.

The algorithm looks for dependencies leaving the segment, and if multiple outgoing edges are found, a couple of simple heuristics are applied. We found that the best performance is achieved when selecting only one outgoing edge. “Small clauses,” which are split into an object and a predicative complement in the dependency framework, are the only cases where we select two heads.

Table 4 shows the results of the dependency-based evaluation. In the table, the output of the

Algorithm 3 Finding head nodes in a segment.

```
input Argument segment $a$
if $a$ overlaps with a segment in the gold standard
    $a \leftarrow$ intersection of $a$ and gold standard
$F \leftarrow \{n :$ governor of $n$ outside $a\}$
if $|F| = 1$
    return $F$
remove punctuation nodes from $F$
if $|F| = 1$
    return $F$
if $F = \{n_1, n_2, \ldots\}$ where $n_1$ is an object and $n_2$ is
  the predicative part of a small clause
    return $\{n_1, n_2\}$
if $F$ contains a node $n$ that is a subject or an object
    return $\{n\}$
else
    return $\{n\}$, where $n$ is the leftmost node in $F$
```

dependency-based system is compared to the semantic dependency links automatically extracted from the segments of the Punyakanok system.

WSJ	P	R	F1	PP
Our system	**88.46**	**83.55**	**85.93**	**61.97**
Punyakanok	87.25	81.59	84.32	58.17

Brown	P	R	F1	PP
Our system	77.67	**69.63**	73.43	**41.32**
Punyakanok	**80.29**	68.59	**73.98**	37.28

WSJ+Brown	P	R	F1	PP
Our system	**87.07**	**81.68**	**84.29**	**59.22**
Punyakanok	86.94	80.21	83.45	55.39

Table 4: Dependency-based evaluation.

In this evaluation, the dependency-based system has a higher F1-measure than the Punyakanok system on both test sets. This suggests that the main advantage of using a dependency-based semantic role labeler is that it is better at finding the heads of semantic arguments, rather than finding segments. The results are also interesting in comparison to the multi-view system described by Pradhan et al. (2005), which has a reported head F1 measure of 85.2 on the WSJ test set. The figure is not exactly compatible with ours, however, since that system used a different head extraction mechanism.

4 Conclusion

We have described a dependency-based system[1] for semantic role labeling of English in the PropBank framework. Our evaluations show that the performance of our system is close to the state of the art. This holds regardless of whether a segment-based or a dependency-based metric is used. Interestingly, our system has a complete proposition accuracy that surpasses other systems by nearly 3 percentage points. Our system is the first semantic role labeler based only on syntactic dependency that achieves a competitive performance.

Evaluation and comparison is a difficult issue since the natural output of a dependency-based system is a set of semantic links rather than segments, as is normally the case for traditional systems. To handle this situation fairly to both types of systems, we carried out a two-way evaluation: conversion of dependencies to segments for the dependency-based system, and head-finding heuristics for segment-based systems. However, the latter is difficult since no structure is available inside segments, and we had to resort to computing upper-bound results using gold-standard input; despite this, the dependency-based system clearly outperformed the upper bound of the performance of the segment-based system. The comparison can also be slightly misleading since the dependency-based system was optimized for the dependency metric and previous systems for the segment metric.

Our evaluations suggest that the dependency-based SRL system is biased to finding argument heads, rather than argument text snippets, and this is of course perfectly logical. Whether this is an advantage or a drawback will depend on the application – for instance, a template-filling system might need complete segments, while an SRL-based vector space representation for text categorization, or a reasoning application, might prefer using heads only.

In the future, we would like to further investigate whether syntactic and semantic analysis could be integrated more tightly. In this work, we used a sim-

[1]Our system is freely available for download at `http://nlp.cs.lth.se/lth_srl`.

plistic loose coupling by means of reranking a small set of complete structures. The same criticisms that are often leveled at reranking-based models clearly apply here too: The set of tentative analyses from the submodules is too small, and the correct analysis is often pruned too early. An example of a method to mitigate this shortcoming is the forest reranking by Huang (2008), in which complex features are evaluated as early as possible.

A Classifier Features

Features Used in Predicate Disambiguation

PREDWORD, PREDLEMMA. The lexical form and lemma of the predicate.

PREDPARENTWORD and PREDPARENTPOS. Form and part-of-speech tag of the parent node of the predicate.

CHILDDEPSET, CHILDWORDSET, CHILDWORDDEPSET, CHILDPOSSET, CHILDPOSDEPSET. These features represent the set of dependents of the predicate using combinations of dependency labels, words, and parts of speech.

DEPSUBCAT. Subcategorization frame: the concatenation of the dependency labels of the predicate dependents.

PREDRELTOPARENT. Dependency relation between the predicate and its parent.

Features Used in Argument Identification and Labeling

PREDLEMMASENSE. The lemma and sense number of the predicate, e.g. *give.01*.

VOICE. For verbs, this feature is Active or Passive. For nouns, it is not defined.

POSITION. Position of the argument with respect to the predicate: Before, After, or On.

ARGWORD and ARGPOS. Lexical form and part-of-speech tag of the argument node.

LEFTWORD, LEFTPOS, RIGHTWORD, RIGHTPOS. Form/part-of-speech tag of the leftmost/rightmost dependent of the argument.

LEFTSIBLINGWORD, LEFTSIBLINGPOS. Form/part-of-speech tag of the left sibling of the argument.

PREDPOS. Part-of-speech tag of the predicate.

RELPATH. A representation of the complex grammatical relation between the predicate and the argument. It consists of the sequence of dependency relation labels and link directions in the path between predicate and argument, e.g. `IM↑OPRD↑OBJ↓`.

VERBCHAINHASSUBJ. Binary feature that is set to true if the predicate verb chain has a subject. The purpose of this feature is to resolve verb coordination ambiguity as in Figure 3.

CONTROLLERHASOBJ. Binary feature that is true if the link between the predicate verb chain and its parent is OPRD, and the parent has an object. This feature is meant to resolve control ambiguity as in Figure 4.

FUNCTION. The grammatical function of the argument node. For direct dependents of the predicate, this is identical to the RELPATH.

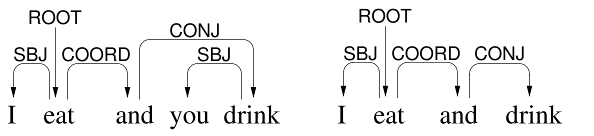


Figure 3: Coordination ambiguity: The subject *I* is in an ambiguous position with respect to *drink*.

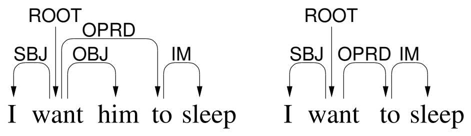


Figure 4: Subject/object control ambiguity: *I* is in an ambiguous position with respect to *sleep*.

References

Collin Baker, Michael Ellsworth, and Katrin Erk. 2007. SemEval task 19: Frame semantic structure extraction. In *Proceedings of SemEval-2007*.

Xavier Carreras and Lluís Màrquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of CoNLL-2004*.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL-2005*.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of CoNLL-2007*.

David M. Chickering, Dan Geiger, and David Heckerman. 1994. Learning Bayesian networks: The combination of knowledge and statistical data. Technical Report MSR-TR-94-09, Microsoft Research.

Ronan Collobert and Jason Weston. 2007. Fast semantic extraction using a novel neural network architecture. In *Proceedings of ACL-2007*.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Schwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *JMLR*, 2006(7):551–585.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of ICCL*.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Daniel Gildea and Martha Palmer. 2002. The necessity of syntactic parsing for predicate argument recognition. In *Proceedings of the ACL-2002*.

Aria Haghighi, Andrew Y. Ng, and Christopher D. Manning. 2005. Robust textual inference via graph matching. In *Proceedings of EMNLP-2005*.

Andrew Hickl, Jeremy Bensley, John Williams, Kirk Roberts, Bryan Rink, and Ying Shi. 2006. Recognizing textual entailment with LCC's GROUNDHOG systems. In *Proceedings of the Second PASCAL Recognizing Textual Entailment Challenge*.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-2008*.

Richard Johansson and Pierre Nugues. 2008a. Dependency-based syntactic–semantic analysis with PropBank and NomBank. In *Proceedings of the Shared Task Session of CoNLL-2008*.

Richard Johansson and Pierre Nugues. 2008b. The effect of syntactic representation on semantic role labeling. In *Proceedings of COLING-2008*.

Chih-Jen Lin, Ruby C. Weng, and S. Sathiya Keerthi. 2008. Trust region Newton method for large-scale logistic regression. *JMLR*, 2008(9):627–650.

Ken Litkowski. 2004. Senseval-3 task: Automatic labeling of semantic roles. In *Proceedings of Senseval-3*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL-2006*.

Alessandro Moschitti, Paul Morărescu, and Sanda Harabagiu. 2003. Open domain information extraction via automatic semantic labeling. In *Proceedings of FLAIRS*.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of ACL-2005*.

Jacob Persson, Richard Johansson, and Pierre Nugues. 2008. Text categorization using predicate–argument structures. Submitted.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Daniel Jurafsky. 2005. Semantic role labeling using different syntactic views. In *Proceedings of ACL-2005*.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of ACL-2003*.

Mihai Surdeanu, Lluís Màrquez, Xavier Carreras, and Pere R. Comas. 2007. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research*, 29:105–151.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL–2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of CoNLL–2008*.

Reid Swanson and Andrew S. Gordon. 2006. A comparison of alternative parse tree paths for labeling semantic roles. In *Proceedings of COLING/ACL-2006*.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2003. Max-margin Markov networks. In *Proceedings of NIPS-2003*.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of ACL-2005*.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP-2004*.

Scaling Textual Inference to the Web

Stefan Schoenmackers, Oren Etzioni, and Daniel S. Weld
Turing Center
University of Washington
Computer Science and Engineering
Box 352350
Seattle, WA 98195, USA
stef,etzioni,weld@cs.washington.edu

Abstract

Most Web-based Q/A systems work by finding pages that contain an *explicit* answer to a question. These systems are helpless if the answer has to be inferred from multiple sentences, possibly on different pages. To solve this problem, we introduce the HOLMES system, which utilizes *textual inference* (TI) over tuples extracted from text.

Whereas previous work on TI (*e.g.*, the literature on textual entailment) has been applied to paragraph-sized texts, HOLMES utilizes knowledge-based model construction to scale TI to a corpus of 117 million Web pages. Given only a few minutes, HOLMES doubles recall for example queries in three disparate domains (geography, business, and nutrition). Importantly, HOLMES's runtime is *linear* in the size of its input corpus due to a surprising property of many textual relations in the Web corpus—they are "approximately" functional in a well-defined sense.

1 Introduction and Motivation

Numerous researchers have identified the Web as a rich source of answers to factual questions, *e.g.*, (Kwok et al., 2001; Brill et al., 2002), but often the desired information is not stated *explicitly* even in a textual corpus as massive as the Web. Consider the question "What vegetables help prevent osteoporosis?" Since there is likely no sentence on the Web directly stating "Kale prevents osteoporosis", a system must *infer* that kale is an answer by combining facts from multiple sentences, possibly from different pages, which justify that conclusion: *i.e.*, that kale is a vegetable, kale contains calcium, and calcium helps prevent osteoporosis.

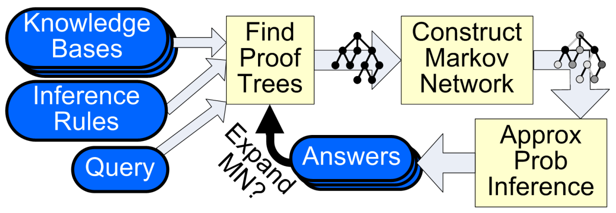


Figure 1: The architecture of HOLMES.

Textual Inference (TI) methods have advanced in recent years. For example, textual entailment techniques aim to determine whether one textual fragment (the *hypothesis*) follows from another (the *text*) (Dagan et al., 2005). While most TI researchers have focused on high-quality inferences from a small source text, we seek to utilize sizable chunks of the Web corpus as our source text. In order to do this, we must confront two major challenges. The first is uncertainty: TI is an imperfect process, particularly when applied to the Web corpus, hence probabilistic methods help to assess the confidence in inferences. The second challenge is scalability: how does inference time scale given increasingly large corpora as input?

1.1 HOLMES: A Scalable TI System

This paper describes HOLMES, an implemented system, which addresses both challenges by carrying out scalable, probabilistic inference over ground assertions extracted from the Web. The input to HOLMES is a conjunctive query, a set of inference rules expressed as Horn clauses, and large sets of ground assertions extracted from the Web, WordNet, and other knowledge bases. As shown in Figure 1, HOLMES chains backward from the query, using the inference rules to construct a forest of proof trees from the ground assertions. This forest is converted into a Markov network (a form of Knowledge-Based Model Construction (KBMC) (Wellman et al., 1992)) and evaluated using approximate probabilistic inference. HOLMES operates in an *anytime* fashion — if desired it can keep iterating: searching for more proofs, and elaborating the Markov network.

HOLMES makes some important simplifying assumptions. Specifically, we use simple ground tuples to represent extracted assertions (*e.g.*, `contains(kale, calcium)`). Syntactic problems (*e.g.*, anaphora, relative clauses) and semantic challenges (*e.g.*, quantification, counterfactuals, temporal qualification) are delegated to the extraction system or simply ignored. This paper focuses on scalability for this subset of the TI task.

1.2 Summary of Experimental Results

We tested HOLMES on 183 million distinct ground assertions extracted from the Web by the TEXTRUNNER system (Banko et al., 2007), coupled with 159 thousand ground assertions from WordNet (Miller et al., 1990), and a compact set of hand-coded inference rules. Given a total of 55 to 145 seconds, HOLMES was able to produce high-quality inferences that doubled the number of answers to example queries in three disparate domains: geography, business, and nutrition.

We also evaluated how the speed of HOLMES scaled with the size of its input corpus. In the general case, logical inference over a Horn theory (needed in order to produce the probabilistic network) is polynomial in the number of ground assertions, and hence in the size of the textual corpus.[1] Unfortunately, this is prohibitive, since even low-order polynomial growth is fatal on a 117 million-page corpus, let alone the full Web.

1.3 Why HOLMES Scales Linearly

Fortunately, the Web's long tail works in our favor. The relations we extract from text are *approximately pseudo-functional* (APF), as we formalize in Section 3, and this property leads to runtime that scales *linearly* with the corpus. To see the underlying intuition, consider the APF relation denoted by the phrase "is married to;" most of the time it maps a person's name to a small number of spousal names so this relation is APF. Section 3 shows why this APF property ensures linear scaling, and Section 4 demonstrates linear scaling in practice.

[1] In fact, it is P-complete — as hard as any polynomial-time problem.

2 An Overview of HOLMES

HOLMES is a system designed to answer complex queries over large, noisy knowledge bases. As a motivating example, we consider the question "What vegetables help prevent osteoporosis?" As of this writing, Google has no pages explicitly stating 'kale helps prevent osteoporosis', making it challenging to return "kale" as an answer. However, there are numerous web pages stating that "kale is high in calcium" and others declaring that "calcium helps prevent osteoporosis". If we could combine those facts we could easily infer that "kale" is an answer to the question "What vegetables help prevent osteoporosis?" HOLMES was designed to make such inferences while accounting for uncertainty in the process.

Given a query, expressed as a conjunctive Datalog rule, HOLMES generates a probabilistic model using knowledge-based model construction (KBMC) (Wellman et al., 1992). Specifically, HOLMES utilizes fast, logical inference to find the subset of ground assertions and inference rules that may influence the answers to the query — enabling the construction of a small and focused Markov network. Since this graphical model is much smaller than one incorporating *all* ground assertions, probabilistic inference will be much faster than if naive compilation were used.

Figure 1 summarizes the operation of HOLMES. As with many theorem provers or KBMC systems, HOLMES takes three inputs:

1. A set of knowledge bases — databases of ground relational assertions, each with an estimate of its probability, which can be generated by TextRunner (Banko et al., 2007) or Kylin (Wu and Weld, 2007). In our example, we would extract the assertions `IsHighIn(kale, calcium)` and `Prevents(calcium, osteoporosis)` from those sentences.
2. A domain theory – A set of probabilistic inference rules written as Markov logic Horn clauses, which can be used to derive new assertions. The weight associated with each clause specifies its reliability.

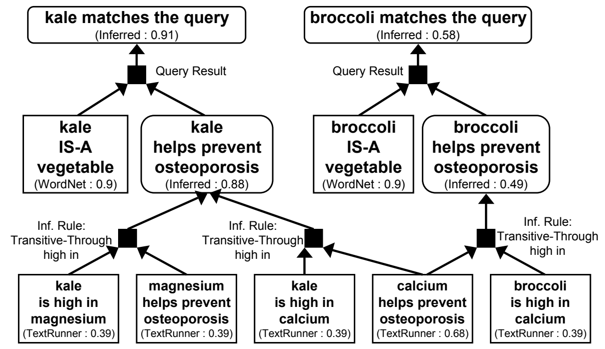

kale matches the query
(Inferred : 0.91)
Query Result
broccoli matches the query
(Inferred : 0.58)
Query Result
kale IS-A vegetable
(WordNet : 0.9)
kale helps prevent osteoporosis
(Inferred : 0.88)
broccoli IS-A vegetable
(WordNet : 0.9)
broccoli helps prevent osteoporosis
(Inferred : 0.49)
Inf. Rule: Transitive-Through high in
Inf. Rule: Transitive-Through high in
Inf. Rule: Transitive-Through high in
kale is high in magnesium
(TextRunner : 0.39)
magnesium helps prevent osteoporosis
(TextRunner : 0.39)
kale is high in calcium
(TextRunner : 0.39)
calcium helps prevent osteoporosis
(TextRunner : 0.68)
broccoli is high in calcium
(TextRunner : 0.39)

Figure 2: Partial proof 'tree' (DAG) for the query "What vegetables help prevent osteoporosis?" Rectangles depict ground assertions from a knowledge base, rounded boxes are inferred assertions, and shaded squared represent the application of inference rules. HOLMES converts this DAG into a Markov network in order to estimate the probability of each node.

In Section 2.3 we identify several domain-independent rules, but a user may (optionally) specify additional, domain-specific rules if desired. In our example, we assume we are given the domain-specific rule: `Prevents(X,Z) :- IsHighIn(X,Y) ∧ Prevents(Y,Z)`

3. A conjunctive query is specified as a Datalog rule. For example, the question "What vegetables help prevent osteoporosis?" could be written as: `query(X) :- IS-A(X,Vegetable) ∧ Prevents(X,osteoporosis)`

and returns a set of answers to the query, each with an associated probability.

2.1 Basic Operation

To find these answers and their associated probabilities, HOLMES first finds all ground assertions in the knowledge bases that are potentially relevant to the query. This is efficiently done using the inference rules to chain backwards from the query. Note that the generated candidate answers, themselves, are less important than the associated proof trees. Furthermore, since HOLMES uses these 'trees' (actually, DAGs) to generate a probabilistic graphical model, HOLMES seeks to find as many proof trees as possible for each query result — each may influence the final belief in that result. Figure 2 shows a partial proof tree for our example query.

To handle uncertainty, HOLMES now constructs a ground Markov network from the proof trees and the Markov-logic-encoded inference rules. Markov networks (Pearl, 1988) model the joint distribution of a set of variables by creating an undirected graph with one node for each random variable, and representing dependencies between variables with cliques in the graph. Each clique has a corresponding potential function ϕ_k, which returns a non-negative value based on the state of variables in the clique. The probability of a state, x, is given by

$$P(x) = \frac{1}{Z} \prod \phi_k(x_{\{k\}})$$

where the partition function Z is a normalizing term, and $x_{\{k\}}$ denotes the state of all the variables in clique k.

HOLMES converts the proof trees into a Markov network in a manner pioneered by the Markov Logic framework of Richardson and Domingos (2006). A Boolean variable is created to represent the truth of each assertion in the proof forest. Next, HOLMES adds edges to the Markov network to create a clique corresponding to each application of an inference rule in the proof forest.

Following the Markov Logic framework, the potential function of a clique has form $\phi(x) = e^w$ if all member nodes are true (w denotes the weight of the inference rule), and $\phi(x) = 1$ otherwise. The probabilities of leaf nodes are derived from the underlying knowledge base,[2] and inferred nodes are biased with an exponential prior.

Finally, HOLMES computes the approximate probability of each answer by running a variant of loopy belief propagation (Pearl, 1988) over the Markov network. In our experience this method performs well on networks derived from our Horn clause proof forest, but one could use Monte Carlo techniques or even exact methods if desired.

Note that this architecture allows HOLMES to combine information from *multiple* web pages to infer assertions not explicitly seen in the textual corpus. Because this inference is done using a Markov network, it correctly handles uncertain extractions and probabilistic dependencies. By using KBMC to create a custom, focused network for each query, the

[2]In our experiments, ground assertions from WordNet get a uniformly high probability of correctness (0.9), but those extracted from the Web are assigned probabilities derived from redundancy statistics, following the intuition that frequently extracted facts are more likely to be true (Etzioni et al., 2005).

amount of probabilistic inference is reduced to manageable proportions.

2.2 Anytime, Incremental Expansion

Because exact probabilistic inference is #P-complete, HOLMES uses approximate methods, but even these techniques have problems if the Markov network gets too large. As a result, HOLMES creates the network incrementally. After the first proof trees are generated, HOLMES creates the model and performs approximate probabilistic inference. If more time is available then HOLMES searches for additional proof trees and updates the network (Figure 1). This incremental process allows HOLMES to return initial results (with preliminary probability estimates) as soon as they are discovered.

For efficiency, HOLMES exploits standard Datalog optimizations (*e.g.*, it only expands proofs of recently added nodes and it uses an approximation to magic sets (Ullman, 1989), rather than simple backwards chaining). For tractability, we also allow the user to limit the number of transitive inference steps for any inference rule.

HOLMES also includes a few enhancements for dealing with information extracted from natural language. For example, HOLMES's inference rules support substring/regex matching of ground assertions, to accommodate simple variations in text. HOLMES also can be restricted to only operate over proper nouns, which is useful for queries involving named entities.

2.3 Markov Logic Inference Rules

HOLMES is given the following set of six domain-independent rules, which are similar to the upward monotone rules introduced by (MacCartney and Manning, 2007).

1. Observed relations are likely to be true:
   `R(X,Y) :- ObservedInCorpus(X, R, Y)`
2. Synonym substitution preserves meaning:
   R_{TR}`(X',Y) :-` R_{TR}`(X,Y)` $\wedge$ `Synonym(X, X')`
3. R_{TR}`(X,Y') :-` R_{TR}`(X,Y)` $\wedge$ `Synonym(Y, Y')`
4. Generalizations preserve meaning:
   R_{TR}`(X',Y) :-` R_{TR}`(X,Y)` $\wedge$ `IS-A(X, X')`
5. R_{TR}`(X,Y') :-` R_{TR}`(X,Y)` $\wedge$ `IS-A(Y, Y')`
6. Transitivity of Part Meronyms:
   R_{TR}`(X,Y') :-` R_{TR}`(X,Y)` $\wedge$ `Part-Of(Y, Y')`
   where R_{TR} matches '* in' (*e.g.*, 'born in').

For example, if `Q(X):-BornIn(X,'France')`, and we know from WordNet that Paris is in France, then by inference rule 6, we know that `BornIn(X,'Paris')` will yield valid results for `Q(X)`. Although all of these rules contain at most two relations in the body, HOLMES allows an arbitrary number of relations in the query and rule bodies. However, we have found that even simple rules can dramatically improve some queries.

We set the rule weights to capture the intuition that deeper inferences decrease the likelihood (as there are more chances to make mistakes), whereas additional, independent proof trees increase the likelihood (as there is more supporting evidence). Specifically, in our experiments we set the prior on inferred facts to -0.75, the weight on rule 1 to 1.5, and the weights on all other rules to 0.6.

At present, we define these weights manually, but we expect to learn the parameter values in the future.

3 Scaling Inference to the Web

If TI is applied to a corpus containing hundreds of millions or even billions of pages, its run time has to be at most linear in the size of the corpus. This section shows that under some reasonable assumptions inference *does* scale linearly.

We start our analysis with two simplifications. First, we assume that the number of distinct, ground assertions in the KBs, $|A|$, grows at most linearly with the size of the textual corpus. This is certainly true for assertions extracted by TextRunner and Kylin, and follows from our exclusion of texts with complex quantified sentences. Our analysis now proceeds to consider scaling with respect to $|A|$ for a fixed query and set of inference rules.

Our second assumption is that the size of every proof tree is bounded by some constant, m. This is a strong assumption and one that depends on the precise set of inference rules and pattern of ground assertions. However, it holds in our experience, and if necessary could be enforced by terminating the search for proof trees at a certain depth, *e.g.*, $\log(m)$.

HOLMES's knowledge-based model construction has two parts: construction of the proof forest and conversion of the forest into a Markov network. Since the Markov network is essentially isomorphic to the proof forest, the conversion will be $\mathcal{O}(|A|)$ if the forest is linear in size, which is ensured if the time to construct the proof trees is $\mathcal{O}(|A|)$. We show this in the remainder of this section.

Recall that HOLMES requires inference rules to be function-free Horn clauses. While this limits expressivity to some degree, it provides a huge speed benefit — logical inference over Horn clauses can be done in polynomial time, whereas general propositional inference (*i.e.*, from grounded first-order rules) is NP-complete.

Alas, even low-order polynomial blowup is unacceptable when the textual corpus reaches Web scale; we seek linear growth. Intuitively, there are two places where polynomial expansion could cause trouble. First, the number of different *types* of proofs (*i.e.*, first order proofs) could grow too quickly, and secondly, a given type of proof tree might apply to too many ground assertions ("tuples" in database lingo). We treat these issues in turn.

Under our assumptions, each proof tree can be represented as an expression in relational algebra with at most m equijoins (Ullman, 1989),[3] each stemming from the application of an inference rule. Since the number of rules is fixed, as is m, there are a constant number of possible first-order proof trees.

The bigger concern is that any one of these first-order trees might result in a polynomial number of ground trees; if so, the size of the ground forest (and corresponding Markov network) could grow too quickly. In fact, polynomial growth is a common phenomena in database query evaluation. Luckily, most relations in the Web corpus behave more favorably. We introduce a property of relations that ensures m-way joins, and therefore all proof trees up to size m, can be computed in $\mathcal{O}(|A|)$ time.

The intuition is that most relations derived from large corpora have a 'heavy-tailed' distribution, wherein a few objects appear many times in a relation, but most appear only once or twice, thus joins involving rare objects lead to a small number of results, and so the main limitation on scalability is common objects. We now prove that if these common objects account for a small enough fraction of the relation, then joins will still scale linearly. We focus on binary relations, but these results can easily be extended to relations of larger arity.

[3]Note that an inference rule of the form `H(X) :- R1(X,Y),R2(Y,Z)` is equivalent to the algebraic expression $\pi_X(R_1 \bowtie R_2)$. First a join is performed between R_1 and R_2 testing for equality between values of Y; then a projection eliminates all columns besides X.

Definition 1 *A relation,* $R = \{(x_i, y_i)\} \subseteq \mathcal{X} \times \mathcal{Y}$, *is* pseudo-functional *(PF)* in x with degree k, *if* $\forall x \in \mathcal{X} : |\{y|(x,y) \in R\}| \leq k$. *When the precise variable and degree is irrelevant to discussion, we simply say "R is PF."*

An m-way equijoin over relations that are PF in the join variables will have at most $k^m * |R|$ results. Since k^m is constant for a given join and $|R|$ scales linearly in the size of the textual corpus, proof tree construction over PF relations also scales linearly.

However, due to their heavy-tailed distributions, most relations extracted from the Web fit the pseudo-functional definition in most, but not all values of $\mathcal{X}$. Fortunately, it turns out that in most cases these "bad" values of $\mathcal{X}$ are rare and hence don't influence the join size significantly. We formalize this intuition by defining a class of approximately pseudo-functional (APF) relations and proving that joining two APF relations produces at most a linear number of results.

Definition 2 *A relation, R, is* approximately pseudo-functional *(APF)* in x with degree k, *if* $\mathcal{X}$ *can be partitioned into two sets* $\mathcal{X}_G$ *and* $\mathcal{X}_B$ *such that for all* $x \in \mathcal{X}_G$ *R is PF with degree k and*

$$\sum_{x \in \mathcal{X}_B} |\{y|(x,y) \in R\}| \leq k * log(|R|)$$

Theorem 1. *If relation* R_1 *is APF in* y *with degree* k_1 *and* R_2 *is APF in* y *with degree* k_2 *then the relation* $Q = R_1 \bowtie R_2$ *has size at most* $\mathcal{O}(max(|R_1|, |R_2|))$.

Proof. Since R_1 and R_2 are APF, we know that $\mathcal{Y}$ can be partitioned into four groups: $\mathcal{Y}_{BB} = \mathcal{Y}_{B1} \bigcap \mathcal{Y}_{B2}$, $\mathcal{Y}_{BG} = \mathcal{Y}_{B1} \bigcap \mathcal{Y}_{G2}$, $\mathcal{Y}_{GB} = \mathcal{Y}_{G1} \bigcap \mathcal{Y}_{B2}$, $\mathcal{Y}_{GG} = \mathcal{Y}_{G1} \bigcap \mathcal{Y}_{G2}$.[4] We can show that each group leads to at most $\mathcal{O}(|A|)$ entries in Q. For $y \in \mathcal{Y}_{BB}$ there are at most $k_1 * k_2 * log(|R_1|) * log(|R_2|)$ entries in Q. The $y \in \mathcal{Y}_{GB}$ and $y \in \mathcal{Y}_{BG}$ lead to at most $k_1 * k_2 * log(|R_2|)$ and $k_1 * k_2 * log(|R_1|)$ entries, respectively. For $y \in \mathcal{Y}_{GG}$ there are at most $k_1 * k_2 * max(|R_1|, |R_2|)$. Summing the results from the four partitions, we see that $|Q|$ is $\mathcal{O}(max(|R_1|, |R_2|))$, thus it is $\mathcal{O}(|A|)$. □

This theorem and proof can easily be extended to

[4]$\mathcal{Y}_{BB}$ are the "doubly bad" values of y that violate the PF definition for both relations, $\mathcal{Y}_{GG}$ are the values that do not violate the PF definition for either relation, and $\mathcal{Y}_{BG}$ and $\mathcal{Y}_{GB}$ are the values that violate it in only R_1 or R_2, resp.

an m-way equijoin, as long as each relation is APF in all arguments that are being joined.

Theorem 2. *If Q is the relation obtained by an equijoin over m relations $R_{1..m}$, each having size at most $\mathcal{O}(|A|)$, and if all $R_{1..m}$ are APF in all arguments that they are joined in with degree at most k_{max}, and if $\prod_{1 \le i \le m} log(|R_i|) \le |A|$, then $|Q|$ is $\mathcal{O}(|A|)$.*

The inequality in Theorem 2 relates the sizes of the relations ($|R|$), the join (m) and the number of ground assertions ($|A|$). However, in many cases we are interested in much smaller values of m than the inequality enables. We can relax the APF definition to allow a broader, but still scalable, class of m-way-APF relations.

Corollary 3. *If Q is the relation obtained by an m-way join, and if each participating relation is APF in their joined variables with a bound of $k_i * \sqrt[m]{|R_i|}$ instead of $k_i * log(|R_i|)$, then the join is $\mathcal{O}(|A|)$.*

The final step in our scaling argument concerns probabilistic inference, which is #P-Complete if performed exactly. This is addressed in two ways. First, HOLMES uses approximate methods, *e.g.*, loopy belief propagation, which avoids the cost of exact inference — at the cost of reduced precision. Secondly, at a practical level, HOLMES's incremental construction of the graphical model (Figure 1) allows it to bound the size of the network by terminating the search for additional proofs.

4 Experimental Results

This section reports on measurements that confirm that linear scaling with $|A|$ occurs in practice, and that HOLMES's inference is not only scalable but also improves precision/recall on sample queries in a diverse set of domains. After describing the experimental domains and queries, Section 4.2 reports on the boost to the area under the precision/recall curve for a set of example queries in three domains: geography, business, and nutrition. Section 4.3 then shows that APF relations are very common in the Web corpus, and finally Section 4.4 demonstrates empirically that HOLMES's inference time scales linearly with the number of pages in the corpus.

4.1 Experimental Setup

HOLMES utilized two knowledge bases in these experiments: TEXTRUNNER and WordNet. TEXTRUNNER contains approximately 183 million distinct ground assertions extracted from over 117 million web pages, and WordNet contains 159 thousand manually created IS-A, Part-Of, and Synonym assertions.

In all queries, HOLMES utilizes the domain-independent inference rules described in Section 2.3. HOLMES additionally makes use of two domain-specific inference rules in the Nutrition domain, to demonstrate the benefits of including domain-specific information. Estimating the precision and relative recall of HOLMES requires extensive and careful manual tagging of HOLMES output. To make this feasible, we restricted ourselves to a set of twenty queries in three domains, but made the domains diverse to illustrate the broad scope of the system.

We now describe each domain briefly.

Geography: the query issued is: "Who was born in one of the following countries?" More formally, `Q(X) :- BornIn(X,{country})` where {country} is bound to each of the following nine countries in turn {France, Germany, China, Thailand, Kenya, Morocco, Peru, Columbia, Guatemala}, yielding a total of nine queries.

Because Web text often refers to a person's birth city rather than birth country, this query illustrates how combining an ground assertion (*e.g.*, `BornIn(Alberto Fujimori, Lima)`) with background knowledge (*e.g.*, `LocatedIn(Lima, Peru)`) enables the system to draw new conclusions (*e.g.*, `BornIn(Alberto Fujimori, Peru)`).

Business: we issued the following two queries.

1) Which companies are acquiring software companies? Formally, `Q(X) :- Acquired(X, Y)` $\wedge$ `Develops(Y, 'software')` This query tests HOLMES's ability to scalably join a large number of assertions from multiple pages.

2) Which companies are headquartered in the USA? `Q(X) :- HeadquarteredIn(X, 'USA')` $\wedge$ `IS-A(X, 'company')`

Answering this query comprehensively requires HOLMES to combine a join (over the relations HeadquarteredIn and IS-A) with transitive inference on PartOf (*e.g.*, Seattle is PartOf Washington which is PartOf the USA) and on IS-A (*e.g.*, Microsoft IS-A software company which IS-A company). The IS-A assertions came from both TEXTRUNNER (using patterns from (Hearst, 1992)) and WordNet.

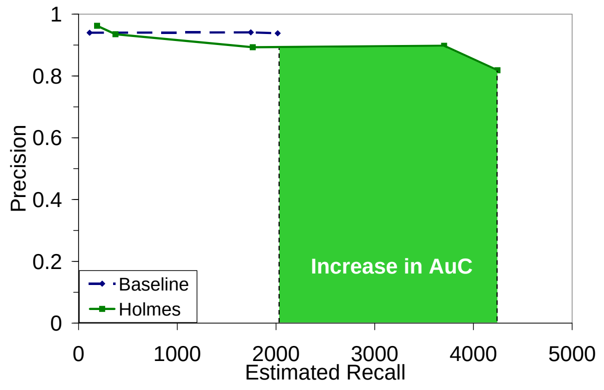


Figure 3: PR Curve for BornIn(X, {country}). Inference boosts the Area under the PR Curve (AuC) by 102 %.

Domain	Increase in AuC	Total Inference Time
Geography	+102%	55 s
Business	+2,643%	145 s
Nutrition	+5,595%	64 s

Table 1: Improvement in the AuC of HOLMES over the BASELINE and total inference time taken by HOLMES. Results are summed over all queries in the geography, business, and nutrition domains. Inference time measured on unoptimized prototype.

Nutrition: the nine queries issued are instances of "What foods prevent disease?" Where a food is a member of one of the classes: fruit, vegetable, or grain, and a disease is one of: anemia, scurvy, or osteoporosis. More formally, `Q(X, {disease}) :- Prevents(X, {disease})` $\wedge$ `IS-A(X, {food})`

Our experiments in the nutrition domain utilized two domain-specific inference rules in addition to the ones presented in Section 2.3:

`Prevents(X,Y):-HighIn(X,Z)` $\wedge$ `Prevents(Z,Y)`

`Prevents(X,Y):-Contains(X,Z)` $\wedge$ `Prevents(Z,Y)`

4.2 Effect of Inference on Recall

To measure the cost and benefit of HOLMES's inference we need to define a baseline for comparison. Answering the conjunctive queries in the business and nutrition domains requires computing joins, which TEXTRUNNER does not do. Thus, we defined a baseline system, BASELINE, which has access to the underlying Knowledge Bases (KBs) (TEXTRUNNER and WordNet), and the ability to compute joins using information explicitly stated in either KB, but does *not* have the ability to infer new assertions.

We compared HOLMES with BASELINE in all three domains. Figure 3 depicts the combined precision/relative recall curves for the nine Geography queries. HOLMES yields substantially higher recall (the shaded region) at modestly lower precision, doubling the area under the precision/recall curve (AuC). The other precision/recall curves also showed a slight drop in precision for substantial gains in recall. Table 1 summarizes the results, along with the total runtime needed for inference. Because relations in the business domain are much larger than in the other domains (*i.e.*, 100x ground assertions), inference is slower in this domain.

We note that inference is particularly helpful with rarely mentioned instances. However, inference can lead to errors when the proof tree contains joins on generic terms (*e.g.*, "company") or common extraction errors (*e.g.*, "LLC" as a company name). This is a key area for future work.

4.3 Prevalence of APF Relations

To determine the prevalence of APF relations in Web text, we examined a sample of 500 binary relations selected randomly from TEXTRUNNER's ground assertions. The surface forms of the relations and arguments may misrepresent the true properties of the underlying concepts, so to better estimate the true properties we merged synonymous values as given by Resolver (Yates and Etzioni, 2007) or the most frequent sense of the word in WordNet. For example, we would consider `BornIn(baby, hospital)` and `BornAt(infant, infirmary)` to represent the same concept, and so would merge them into one instance of the 'Born In' relation. The largest two relations had over 1.25 million unique instances each, and 52% of the relations had more than 10,000 instances.

For each relation R, we first found all instances of R extracted by TEXTRUNNER and merged all synonymous instances as described above. Then, for each argument of R we computed the smallest value, K_{min}, such that R is APF with degree K_{min}. Since many interesting assertions can be inferred by simply joining two relations, we also considered the special case of 2-way joins using Corollary 3. We computed the smallest value, $K_{2\bowtie}$, such that the relation is two-way-APF with degree $K_{2\bowtie}$.

Figure 4 shows the fraction of relations with K_{min} and $K_{2\bowtie}$ of at most K as a function of varying

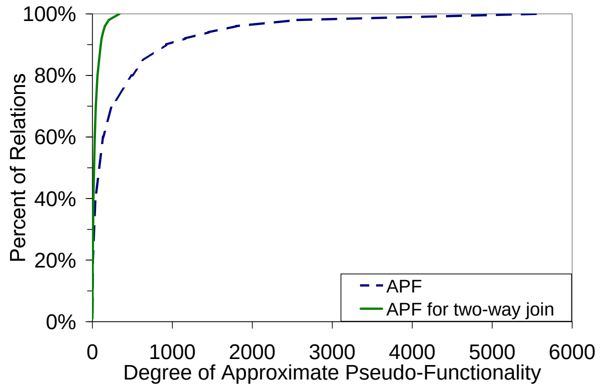


Figure 4: Prevalence of APF relations in Web text. The x-axis depicts the *degree* of pseudo-functionality, *e.g.*, K_{min} and $K_{2\bowtie}$, (see definition 2); the y-axis lists the percent of relations that are APF with that degree. Results are averaged over both arguments.

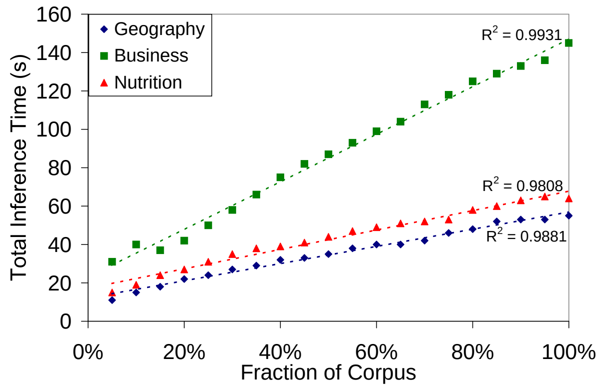


Figure 5: The effects of corpus size on total inference time. We see approximately linear growth in all domains, and display the best fit lines and coefficient of determination (R^2) of each.

values of K. The results are averaged over both arguments of each binary relation. For arbitrary joins in this KB, 80% of the relations are APF with degree less than 496; for 2-way joins (like the ones in our inference rules and test queries), 80% of the relations are APF with degree less than 65. These results indicate that the majority of relations TEXTRUNNER extracted from text are APF, and so we can expect HOLMES's techniques will allow efficient inference over most relations.

While Theorem 2 guarantees that joins over those relations will be $\mathcal{O}(|R|)$, that notation hides a potentially large constant factor of ${K_{min}}^m$. Fortunately the constant factor is significantly smaller in practice. To see why, we re-examine the proof: the large factor comes from assuming that *all* of R's first arguments which meet the PF definition are associated with exactly K_{min} distinct second arguments. However, in our corpus 83% of first arguments are associated with only *one* second argument. Clearly, our worst-case analysis substantially over-estimates inference time for most queries. Moreover, in additional experiments (omitted due to space limitations), measured join sizes grew linearly in the size of the corpus, but were on average two to three orders of magnitude smaller than the bounds given in the theory. This observation held across relations with different sizes and values of K_{min}.

While the results in Figure 4 may vary for other sets of relations, we believe the general trends hold. This is promising for Question Answering and Textual Inference systems, since if true it implies that combining information from multiple difference source is feasible, and can allow such systems to infer answers not explicitly seen in any source.

4.4 Scalability of Inference Speed

Since the previous subsection showed that most relations are APF in their arguments, our theory predicts HOLMES's inference will scale linearly. We tested this hypothesis empirically by running inference over the test queries in our three domains, while varying the number of pages in the textual corpus.

Figure 5 shows how the inference time HOLMES used to answer all queries in each domain scales with KB size. For these queries, and several others we tested (not shown here), inference time grows linearly with the size of the KB. Based on these results we believe that HOLMES can provide scalable inference over a wide variety of domains.

5 Related Work

Textual Entailment systems are given two textual fragments, text T and hypothesis H, and attempt to decide if the meaning of H can be inferred from the meaning of T (Dagan et al., 2005). While many approaches have addressed this problem, our work is most closely related to that of (Raina et al., 2005; MacCartney and Manning, 2007; Tatu and Moldovan, 2006; Braz et al., 2005), which convert the inputs into logical forms and then attempt to 'prove' H from T plus a set of axioms. For instance, (Braz et al., 2005) represents T, H, and a set of rewrite rules in a description logic framework, and determines entailment by solving an integer linear program derived from that representation.

These approaches and related ones (*e.g.*, (Van Durme and Schubert, 2008)) use highly expressive representations, enabling them to express negation, temporal information, and more. HOLMES's representation is much simpler—Markov Logic Horn Clauses for inference rules coupled with a massive database of ground assertions. However, this simplification allows HOLMES to tackle a "text" of enormously larger size: 117 million Web pages versus a single paragraph. A second, if smaller, difference stems from the fact that instead of determining whether a single hypothesis sentence, H, follows from the text, HOLMES tries to find all consequents that match a conjunctive query.

HOLMES is also related to open-domain question-answering systems such as Mulder (Kwok et al., 2001), AskMSR (Brill et al., 2002), and others (Harabagiu et al., 2000; Brill et al., 2001). However, these Q/A systems attempt to find individual documents or sentences containing the answer. They often perform deep analysis on promising texts, and back off to shallower, less reliable methods if those fail. In contrast, HOLMES utilizes TI and attempts to combine information from multiple different sentences in a scalable way.

While its ability to combine information from multiple sources is promising, HOLMES has several limitations these Q/A systems do not have. Since HOLMES relies on an information extraction system to convert sentences into ground predicates, any limitations of the IE system will be propagated to HOLMES. Additionally, the logical representation HOLMES uses limits the reasoning and types of questions it can answer. HOLMES is geared towards answering questions which are naturally expressed as properties and relations of entities, and is not well suited to answering more abstract or open ended questions. Although we have demonstrated that HOLMES is scalable, further work is needed to make it to run at interactive speeds.

Finally, research in statistical relational learning such as MLNs (Richardson and Domingos, 2006), RMNs (Taskar et al., 2002), and others (Getoor and Taskar, 2007) have studied techniques for combining logical and probabilistic inference. Our inference rules are more restrictive than those allowed in MLNs, but this trade-off allows us to efficiently scale inference to large, open domain corpora. By constructing only cliques for satisfied inference rules, HOLMES explicitly models the intuition behind LazySAT inference (Singla and Domingos, 2006) as used in MLNs. *I.e.*, most Horn clause inference rules will be trivially satisfied since their antecedents will be false, so we only need to worry about ones where the antecedent is true.

6 Conclusions

This paper makes three main contributions:

1. We introduce and evaluate the HOLMES system, which leverages KBMC methods in order to scale a class of TI methods to the Web.

2. We define the notion of *Approximately Pseudo-Functional* (APF) relations and prove that, for a APF relations, HOLMES's inference time increases *linearly* with the size of the input corpus. We show empirically that APF relations appear to be prevalent in our Web corpus (Figure 4), and that HOLMES's runtime does scale linearly with the size of its input (Figure 5), taking only a few CPU minutes when run over 183 million distinct ground assertions.

3. We present experiments demonstrating that, for a set of queries in the domains of geography, business, and nutrition, HOLMES substantially improves the quality of answers (measured by AuC) relative to a "no inference" baseline.

In the future, we plan more extensive tests to characterize when HOLMES's inference is helpful. We also hope to examine in what cases jointly performing extraction and inference (as opposed to performing them separately) is feasible at scale. Finally, we plan to examine methods for HOLMES to learn both rule weights and new inference rules.

Acknowledgements

We thank the following for helpful comments on previous drafts: Fei Wu, Michele Banko, Mausam, Doug Downey, and Alan Ritter. This research was supported in part by NSF grants IIS-0535284, IIS-0312988, and IIS-0307906, ONR grants N00014-08-1-0431 and N00014-06-1-0147, CALO grant 03-000225, the WRF / TJ Cable Professorship as well as gifts from Google. The work was performed at the University of Washington's Turing Center.

References

M. Banko, M. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open information extraction from the Web. In *Procs. of IJCAI*.

R. Braz, R. Girju, V. Punyakanok, D. Roth, and M. Sammons. 2005. An inference model for semantic entailment in natural language. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1678–1679.

E. Brill, J. Lin, M. Banko, S. T. Dumais, and A. Y. Ng. 2001. Data-intensive question answering. In *Procs. of Text REtrieval Conference (TREC-10)*, pages 393–400.

Eric Brill, Susan Dumais, and Michele Banko. 2002. An analysis of the AskMSR question-answering system. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 257–264, Morristown, NJ, USA. Association for Computational Linguistics.

I. Dagan, O. Glickman, and B. Magnini. 2005. The PASCAL Recognising Textual Entailment Challenge. *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, pages 1–8.

O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134.

L. Getoor and B. Taskar. 2007. *Introduction to Statistical Relational Learning*. MIT Press.

S. Harabagiu, M. Pasca, and S. Maiorano. 2000. Experiments with open-domain textual question answering. In *Procs. of the COLING-2000.*

M. Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Procs. of the 14th International Conference on Computational Linguistics*, pages 539–545, Nantes, France.

C.C.T. Kwok, O. Etzioni, and D.S. Weld. 2001. Scaling question answering to the Web. *Proceedings of the 10th international conference on World Wide Web*, pages 150–161.

B. MacCartney and C.D. Manning. 2007. Natural Logic for Textual Inference. In *Workshop on Textual Entailment and Paraphrasing*.

G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. 1990. Introduction to wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–312.

Judea Pearl. 1988. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

Rajat Raina, Andrew Y. Ng, and Christopher D. Manning. 2005. Robust textual inference via learning and abductive reasoning. In *Proceedings of AAAI 2005*. AAAI Press.

M. Richardson and P. Domingos. 2006. Markov Logic Networks. *Machine Learning*, 62(1-2):107–136.

Parag Singla and Pedro Domingos. 2006. Memory-efficient inference in relational domains. In *AAAI*.

B. Taskar, P. Abbeel, and D. Koller. 2002. Discriminative probabilistic models for relational data. *Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI02)*.

Marta Tatu and Dan Moldovan. 2006. A logic-based semantic approach to recognizing textual entailment. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 819–826, Morristown, NJ, USA. Association for Computational Linguistics.

J. Ullman. 1989. *Database and knowledge-base systems*. Computer Science Press.

B. Van Durme and L.K. Schubert. 2008. Open knowledge extraction through compositional language processing. In *Symposium on Semantics in Systems for Text Processing*.

M. Wellman, J. Breese, and R. Goldman. 1992. From knowledge bases to decision models. *The Knowledge Engineering Review*, 7(1):35–53.

F. Wu and D. Weld. 2007. Autonomously semantifying Wikipedia. In *Proceedings of the ACM Sixteenth Conference on Information and Knowledge Management (CIKM-07)*, Lisbon, Porgugal.

A. Yates and O. Etzioni. 2007. Unsupervised resolution of objects and relations on the Web. In *Procs. of HLT*.

Maximum Entropy based Rule Selection Model for Syntax-based Statistical Machine Translation

Qun Liu[1] and **Zhongjun He**[1,2] and **Yang Liu**[1] and **Shouxun Lin**[1]
[1]Key Laboratory of Intelligent Information Processing
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, 100190, China
[2]Graduate University of Chinese Academy of Sciences
Beijing, 100049, China
{liuqun,zjhe,yliu,sxlin}@ict.ac.cn

Abstract

This paper proposes a novel maximum entropy based rule selection (MERS) model for syntax-based statistical machine translation (SMT). The MERS model combines local contextual information around rules and information of sub-trees covered by variables in rules. Therefore, our model allows the decoder to perform context-dependent rule selection during decoding. We incorporate the MERS model into a state-of-the-art linguistically syntax-based SMT model, the tree-to-string alignment template model. Experiments show that our approach achieves significant improvements over the baseline system.

1 Introduction

Syntax-based statistical machine translation (SMT) models (Liu et al., 2006; Galley et al., 2006; Huang et al., 2006) capture long distance reorderings by using rules with structural and linguistical information as translation knowledge. Typically, a translation rule consists of a source-side and a target-side. However, the source-side of a rule usually corresponds to multiple target-sides in multiple rules. Therefore, during decoding, the decoder should select a correct target-side for a source-side. We call this *rule selection*.

Rule selection is of great importance to syntax-based SMT systems. Comparing with word selection in word-based SMT and phrase selection in phrase-based SMT, rule selection is more generic and important. This is because that a rule not only contains terminals (words or phrases), but also contains nonterminals and structural information. Terminals indicate lexical translations, while nonterminals and structural information can capture short or long distance reorderings. See rules in Figure 1 for illustration. These two rules share the same syntactic tree on the source side. However, on the target side, either the translations for terminals or the phrase reorderings for nonterminals are quite different. During decoding, when a rule is selected and applied to a source text, both lexical translations (for terminals) and reorderings (for nonterminals) are determined. Therefore, rule selection affects both lexical translation and phrase reordering.

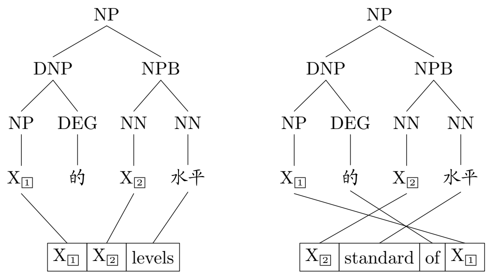


Figure 1: Example of translation rules

However, most of the current syntax-based systems ignore contextual information when they selecting rules during decoding, especially the information of sub-trees covered by nonterminals. For example, the information of $X_{\boxed{1}}$ and $X_{\boxed{2}}$ is not recorded when the rules in Figure 1 extracted from the training examples in Figure 2. This makes the decoder hardly distinguish the two rules. Intuitively, information of sub-trees covered by nonterminals as well as contextual information of rules are believed

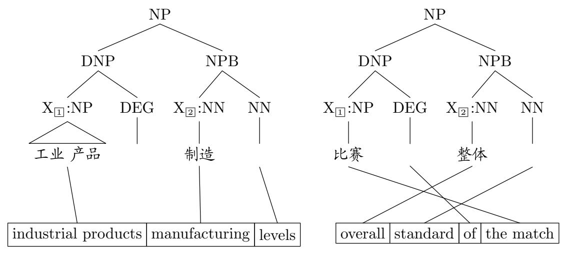


Figure 2: Training examples for rules in Figure 1

to be helpful for rule selection.

Recent research showed that contextual information can help perform word or phrase selection. Carpuat and Wu (2007b) and Chan et al. (2007) showed improvents by integrating word-sense-disambiguation (WSD) system into a phrase-based (Koehn, 2004) and a hierarchical phrase-based (Chiang, 2005) SMT system, respectively. Similar to WSD, Carpuat and Wu (2007a) used contextual information to solve the ambiguity problem for phrases. They integrated a phrase-sense-disambiguation (PSD) model into a phrase-based SMT system and achieved improvements.

In this paper, we propose a novel solution for rule selection for syntax-based SMT. We use the maximum entropy approach to combine rich contextual information around a rule and the information of sub-trees covered by nonterminals in a rule. For each ambiguous source-side of translation rules, a maximum entropy based rule selection (MERS) model is built. Thus the MERS models can help the decoder to perform a context-dependent rule selection.

Comparing with WSD (or PSD), there are some advantages of our approach:

- Our approach resolves ambiguity for rules with multi-level syntactic structure, while WSD resolves ambiguity for strings that have no structures;
- Our approach can help the decoder perform both lexical selection and phrase reorderings, while WSD can help the decoder only perform lexical selection;
- Our method takes WSD as a special case, since a rule may only consists of terminals.

In our previous work (He et al., 2008), we reported improvements by integrating a MERS model into a formally syntax-based SMT model, the hierarchical phrase-based model (Chiang, 2005). In this paper, we incorporate the MERS model into a state-of-the-art linguistically syntax-based SMT model, the tree-to-string alignment template (TAT) model (Liu et al., 2006). The basic differences are:

- The MERS model here combines rich information of source syntactic tree as features since the translation model is linguistically syntax-based. He et al. (2008) did not use this information.
- In this paper, we build MERS models for all ambiguous source-sides, including lexicalized (source-side which only contains terminals), partially lexicalized (source-side which contains both terminals and nonterminals), and unlexicalized (source-side which only contains nonterminals). He et al. (2008) only built MERS models for partially lexicalized source-sides.

In the TAT model, a TAT can be considered as a translation rule which describes correspondence between source syntactic tree and target string. TAT can capture linguistically motivated reorderings at short or long distance. Experiments show that by incorporating MERS model, the baseline system achieves statistically significant improvement.

This paper is organized as follows: Section 2 reviews the TAT model; Section 3 introduces the MERS model and describes feature definitions; Section 4 demonstrates a method to incorporate the MERS model into the translation model; Section 5 reports and analyzes experimental results; Section 6 gives conclusions.

2 Baseline System

Our baseline system is Lynx (Liu et al., 2006), which is a linguistically syntax-based SMT system. For translating a source sentence $f_1^J = f_1 ... f_j ... f_J$, Lynx firstly employs a parser to produce a source syntactic tree $T(f_1^J)$, and then uses the source syntactic tree as the input to search translations:

$$(1)\ \widetilde{e}_1^I = \text{argmax}_{e'^I_1} Pr(e_1^I|f_1^J)$$
$$= \text{argmax}_{e'^I_1} Pr(T(f_1^J)|f_1^J) Pr(e_1^I|T(f_1^J))$$

In doing this, Lynx uses tree-to-string alignment template to build relationship between source syntactic tree and target string. A TAT is actually a translation rule: the source-side is a parser tree with leaves consisting of words and nonterminals, the target-side is a target string consisting of words and nonterminals.

TAT can be learned from word-aligned, source-parsed parallel corpus. Figure 4 shows three types of TATs extracted from the training example in Figure 3: lexicalized (the left), partially lexicalized (the middle), unlexicalized (the right). Lexicalized TAT contains only terminals, which is similar to phrase-to-phrase translation in phrase-based model except that it is constrained by a syntactic tree on the source-side. Partially lexicalized TAT contains both terminals and non-terminals, which can be used for both lexical translation and phrase reordering. Unlexicalized TAT contains only nonterminals and can only be used for phrase reordering.

Lynx builds translation model in a log-linear framework (Och and Ney, 2002):

$$(2)\quad P(e_1^I|T(f_1^J)) = \frac{\exp[\sum_m \lambda_m h_m(e_1^I, T(f_1^J))]}{\sum_{e'} \exp[\sum_m \lambda_m h_m(e_1^I, T(f_1^J))]}$$

Following features are used:

- Translation probabilities: $P(\widetilde{e}|\widetilde{T})$ and $P(\widetilde{T}|\widetilde{e})$;
- Lexical weights: $P_w(\widetilde{e}|\widetilde{T})$ and $P_w(\widetilde{T}|\widetilde{e})$;
- TAT penalty: $exp(1)$, which is analogous to phrase penalty in phrase-based model;
- Language model $P_{lm}(e_1^I)$;
- Word penalty I.

In Lynx, rule selection mainly depends on translation probabilities and lexical weights. These four scores describe how well a source tree links to a target string, which are estimated on the training corpus according to occurrence times of $\widetilde{e}$ and $\widetilde{T}$. There are no features in Lynx that can capture contextual information during decoding, except for the n-gram language model which considers the left and right neighboring n-1 target words. But this information it very limited.

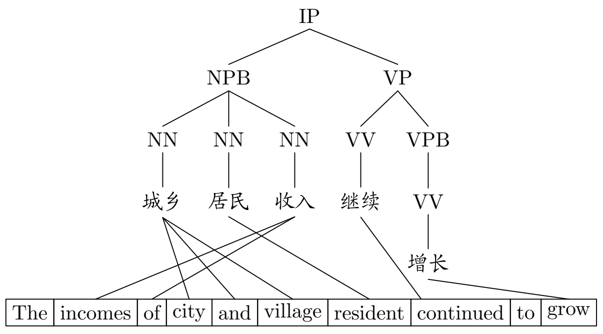


Figure 3: Word-aligned, source-parsed training example.

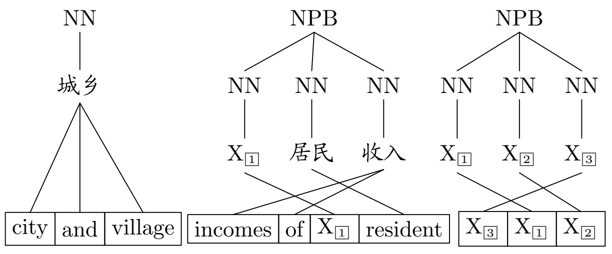


Figure 4: TATs learned from the training example in Figure 3.

3 The Maximum Entropy based Rule Selection Model

3.1 The model

In this paper, we focus on using contextual information to help the TAT model perform context-dependent rule selection. We consider the rule selection task as a multi-class classification task: for a source syntactic tree $\widetilde{T}$, each corresponding target string $\widetilde{e}$ is a label. Thus during decoding, when a TAT $\langle \widetilde{T}, \widetilde{e'} \rangle$ is selected, $\widetilde{T}$ is classified into label $\widetilde{e'}$, actually.

A good way to solve the classification problem is the maximum entropy approach:

$$(3)\quad P_{rs}(\widetilde{e}|\widetilde{T}, T(X_k)) = \frac{\exp[\sum_i \lambda_i h_i(\widetilde{e}, C(\widetilde{T}), T(X_k))]}{\sum_{\widetilde{e'}} \exp[\sum_i \lambda_i h_i(\widetilde{e'}, C(\widetilde{T}), T(X_k))]}$$

where $\widetilde{T}$ and $\widetilde{e}$ are the source tree and target string of a TAT, respectively. h_i is a binary feature functions and λ_i is the feature weight of h_i. $C(\widetilde{T})$ defines local contextual information of $\widetilde{T}$. X_k is a nonterminal in the source tree $\widetilde{T}$, where k is an index. $T(X_k)$ is the source sub-tree covered by X_k.

The advantage of the MERS model is that it uses rich contextual information to compute posterior probability for $\widetilde{e}$ given $\widetilde{T}$. However, the translation probabilities and lexical weights in Lynx ignore these information.

Note that for each ambiguous source tree, we build a MERS model. That means, if there are N source trees extracted from the training corpus are ambiguous (the source tree which corresponds to multiple translations), thus for each ambiguous source tree T_i ($i = 1, ..., N$), a MERS model M_i ($i = 1, ..., N$) is built. Since a source tree may correspond to several hundreds of target translations at most, the feature space of a MERS model is not prohibitively large. Thus the complexity for training a MERS model is low.

3.2 Feature Definition

Let $\langle \widetilde{T}, \widetilde{e} \rangle$ be a translation rule in the TAT model. We use $f(\widetilde{T})$ to represent the source phrase covered by $\widetilde{T}$. To build a MERS model for the source tree $\widetilde{T}$, we explore various features listed below.

1. Lexical Features (LF)
   These features are defined on source words. Specifically, there are two kinds of lexical features: **external** features f_{-1} and f_{+1}, which are the source words immediately to the left and right of $f(\widetilde{T})$, respectively; **internal** features $f_L(T(X_k))$ and $f_R(T(X_k))$, which are the left most and right most boundary words of the source phrase covered by $T(X_k)$, respectively.

   See Figure 5 (a) for illustration. In this example, f_{-1}=tígāo, f_{+1}=zhìzào, $f_L(T(X_1))$=gōngyè, $f_R(T(X_1))$=chǎnpǐn.

2. Parts-of-speech (POS) Features (POSF)
   These features are the POS tags of the source words defined in the lexical features: P_{-1}, P_{+1}, $P_L(T(X_k))$, $P_R(T(X_k))$ are the POS tags of f_{-1}, f_{+1}, $f_L(T(X_k))$, $f_R(T(X_k))$, re-

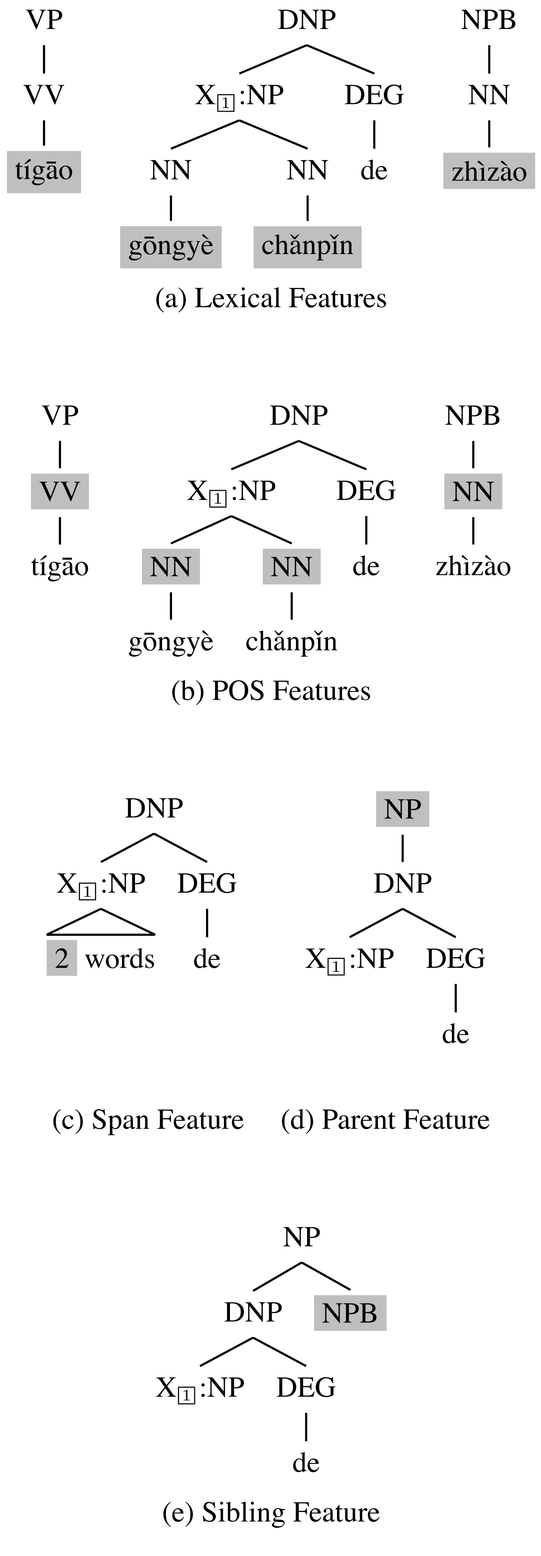


Figure 5: Illustration of features of the MERS model. The source tree of the TAT is ⟨ DNP(NP $X_{\boxed{1}}$) (DEG de)⟩. Gray nodes denote information included in the feature.

spectively. POS tags can generalize over all training examples.

Figure 5 (b) shows POS features. P_{-1}=VV, P_{+1}=NN, $P_L(T(X_1))$=NN, $P_R(T(X_1))$=NN.

3. Span Features (SPF)
These features are the length of the source phrase $f(T(X_k))$ covered by $T(X_k)$. In Liu's TAT model, the knowledge learned from a short span can be used for a larger span. This is not reliable. Thus we use span features to allow the MERS model to learn a preference for short or large span.

   In Figure 5 (c), the span of $X_{\boxed{1}}$ is 2.

4. Parent Feature (PF)
The parent node of $\widetilde{T}$ in the parser tree of the source sentence. The same source sub-tree may have different parent nodes in different training examples. Therefore, this feature may provide information for distinguishing source sub-trees.

   Figure 5 (d) shows that the parent is a *NP* node.

5. Sibling Features (SBF)
The siblings of the root of $\widetilde{T}$. This feature considers neighboring nodes which share the same parent node.

   In Figure 5 (e), the source tree has one sibling node *NPB*.

Those features make use of rich information around a rule, including the contextual information of a rule and the information of sub-trees covered by nonterminals. They are never used in Liu's TAT model.

Figure 5 shows features for a partially lexicalized source tree. Furthermore, we also build MERS models for lexicalized and unlexicalized source trees. Note that for lexicalized tree, features do not include the information of sub-trees since there is no nonterminals.

The features can be easily obtained by modifying the TAT extraction algorithm described in (Liu et al., 2006). When a TAT is extracted from a word-aligned, source-parsed parallel sentence, we just record the contextual features and the features of the sub-trees. Then we use the toolkit implemented by Zhang (2004) to train MERS models for the ambiguous source syntactic trees separately. We set the iteration number to 100 and Gaussian prior to 1.

4 Integrating the MERS Models into the Translation Model

We integrate the MERS models into the TAT model during the translation of each source sentence. Thus the MERS models can help the decoder perform context-dependent rule selection during decoding.

For integration, we add two new features into the log-linear translation model:

- $P_{rs}(\widetilde{e}|\widetilde{T}, T(X_k))$. This feature is computed by the MERS model according to equation (3), which gives a probability that the model selecting a target-side $\widetilde{e}$ given an ambiguous source-side $\widetilde{T}$, considering rich contextual information.

- $P_{ap} = exp(1)$. During decoding, if a source tree has multiple translations, this feature is set to $exp(1)$, otherwise it is set to $exp(0)$. Since the MERS models are only built for ambiguous source trees, the first feature $P_{rs}(\widetilde{e}|\widetilde{T}, T(X_k))$ for non-ambiguous source tree will be set to 1.0. Therefore, the decoder will prefer to use non-ambiguous TATs. However, non-ambiguous TATs usually occur only once in the training corpus, which are not reliable. Thus we use this feature to reward ambiguous TATs.

The advantage of our integration is that we need not change the main decoding algorithm of Lynx. Furthermore, the weights of the new features can be trained together with other features of the translation model.

5 Experiments

5.1 Corpus

We carry out experiments on Chinese-to-English translation. The training corpus is the FBIS corpus, which contains 239k sentence pairs with 6.9M Chinese words and 8.9M English words. For the language model, we use SRI Language Modeling Toolkit (Stolcke, 2002) with modified Kneser-Ney smoothing (Chen and Goodman, 1998) to train two tri-gram language models on the English portion of

Type	No. of TATs	No. of source trees	No. of ambiguous source trees	% ambiguous
Lexicalized	333,077	16,367	14,380	87.86
Partially Lexicalized	342,767	38,497	28,397	73.76
Unlexicalized	83,024	7,384	5,991	81.13
Total	758,868	62,248	48,768	78.34

Table 1: Statistical information of TATs filtered by test sets of NIST MT 2003 and 2005.

System	Features									
	$P(\widetilde{e}\|\widetilde{T})$	$P(\widetilde{T}\|\widetilde{e})$	$P_w(\widetilde{e}\|\widetilde{T})$	$P_w(\widetilde{T}\|\widetilde{e})$	lm_1	lm_2	TP	WP	P_{rs}	AP
Lynx	0.210	0.016	0.081	0.051	0.171	0.013	-0.055	0.403	-	-
+MERS	0.031	0.008	0.020	0.080	0.152	0.014	0.027	0.270	0.194	0.207

Table 2: Feature weights obtained by minimum error rate training on the development set. The first 8 features are used by Lynx. TP=TAT penalty, WP=word penalty, AP=ambiguous TAT penalty. Note that in fact, the positive weight for WP and AP indicate a reward.

the training corpus and the Xinhua portion of the Gigaword corpus, respectively. NIST MT 2002 test set is used as the development set. NIST MT 2003 and NIST MT 2005 test sets are used as the test sets. The translation quality is evaluated by BLEU metric (Papineni et al., 2002), as calculated by mteval-v11b.pl with case-insensitive matching of n-grams, where $n = 4$.

5.2 Training

To train the translation model, we first run GIZA++ (Och and Ney, 2000) to obtain word alignment in both translation directions. Then the word alignment is refined by performing "grow-diag-final" method (Koehn et al., 2003). We use a Chinese parser developed by Deyi Xiong (Xiong et al., 2005) to parse the Chinese sentences of the training corpus.

Our TAT extraction algorithm is similar to Liu et al. (2006), except that we make some tiny modifications to extract contextual features for MERS models. To extract TAT, we set the maximum height of the source sub-tree to $h = 3$, the maximum number of direct descendants of a node of sub-tree to $c = 5$. See (Liu et al., 2006) for specific definitions of these parameters.

Table 1 shows statistical information of TATs which are filtered by the two test sets. For each type (lexicalized, partially lexicalized, unlexicalized) of TATs, a great portion of the source trees are ambiguous. The number of ambiguous source trees accounts for 78.34% of the total source trees. This indicates that the TAT model faces serious rule selection problem during decoding.

5.3 Results

We use Lynx as the baseline system. Then the MERS models are incorporated into Lynx, and the system is called Lynx+MERS. To run the decoder, Lynx and Lynx+MERS share the same settings: tatTable-limit=30, tatTable-threshold=0, stack-limit=100, stack-threshold=0.00001. The meanings of the pruning parameters are the same to Liu et al. (2006).

We perform minimum error rate training (Och, 2003) to tune the feature weights for the log-linear model to maximize the systems's BLEU score on the development set. The weights are shown in Table 2.

These weights are then used to run Lynx and Lynx+MERS on the test sets. Table 3 shows the results. Lynx obtains BLEU scores of 26.15 on NIST03 and 26.09 on NIST05. Using all features described in Section 3.2, Lynx+MERS finally obtains BLEU scores of 27.05 on NIST03 and 27.28 on NIST05. The absolute improvements is 0.90 and 1.19, respectively. Using the sign-test described by Collins et al. (2005), both improvements are statistically significant at $p < 0.01$. Moreover, Lynx+MERS also achieves higher n-gram precisions than Lynx.

Test Set	System	BLEU-4	Individual n-gram precisions			
			1	2	3	4
NIST03	Lynx	26.15	71.62	35.64	18.64	9.82
	+MERS	**27.05**	72.00	36.72	19.51	10.37
NIST05	Lynx	26.09	70.39	35.12	18.53	10.11
	+MERS	**27.28**	71.16	36.19	19.62	10.95

Table 3: BLEU-4 scores (case-insensitive) on the test sets.

5.4 Analysis

The baseline system only uses four features for rule selection: the translation probabilities $P(\widetilde{e}|\widetilde{T})$ and $P(\widetilde{T}|\widetilde{e})$; and the lexical weights $P_w(\widetilde{e}|\widetilde{T})$ and $P_w(\widetilde{T}|\widetilde{e})$. These features are estimated on the training corpus by the maximum likelihood approach, which does not allow the decoder to perform a context dependent rule selection. Although Lynx uses language model as feature, the n-gram language model only considers the left and right n-1 neighboring target words.

The MERS models combines rich contextual information as features to help the decoder perform rule selection. Table 4 shows the effect of different feature sets. We test two classes of feature sets: the single feature (the top four rows of Table 4) and the combination of features (the bottom five rows of Table 4). For the single feature set, the POS tags are the most useful and stable features. Using this feature, Lynx+MERS achieves improvements on both the test sets. The reason is that POS tags can be generalized over all training examples, which can alleviate the data sparseness problem.

Although we find that some single features may hurt the BLEU score, they are useful in combination of features. This is because one of the strengths of the maximum entropy model is that it can incorporate various features to perform classification. Therefore, using all features defined in Section 3.2, we obtain statistically significant improvements (the last row of Table 4). In order to know how the MERS models improve translation quality, we inspect the 1-best outputs of Lynx and Lynx+MERS. We find that the first way that the MERS models help the decoder is that they can perform better selection for words or phrases, similar to the effect of WSD or PSD. This is because that lexicalized and partially lexicalized TAT contains terminals. Considering the following examples:

Feature Sets	NIST03	NIST05
LF	26.12	26.32
POSF	26.36	26.21
PF	26.17	25.90
SBF	26.47	26.08
LF+POSF	26.61	26.59
LF+POSF+SPF	26.70	26.44
LF+POSF+PF	26.81	26.56
LF+POSF+SBF	26.68	26.89
LF+POSF+SPF+PF+SBF	27.05	27.28

Table 4: BLEU-4 scores on different feature sets.

- Source: 马耳他 <u>位于</u> 欧洲 南部
- Reference: Malta <u>is located in</u> southern Europe
- Lynx: Malta <u>in</u> southern Europe
- Lynx+MERS: Malta <u>is located in</u> southern Europe

Here the Chinese word "位 于" is incorrectly translated into "in" by the baseline system. Lynx+MERS produces the correct translation "is located in". That is because, the MERS model considers more contextual information for rule selection. In the MERS model, $P_{rs}(\text{in}|\text{位于}) = 0.09$, which is smaller than $P_{rs}(\text{is located in}|\text{位于}) = 0.14$. Therefore, the MERS model prefers the translation "is located in". Note that here the source tree (VV 位于) is lexicalized, and the role of the MERS model is actually the same as WSD.

The second way that the MERS models help the decoder is that they can perform better phrase reorderings. Considering the following examples:

- Source: 按照 [在 中国 市场]$_1$ 的 [发展 战略]$_2$ ...
- Reference: According to its [development strategy]$_2$ [in the Chinese market]$_1$ ...
- Lynx: Accordance with [the Chinese market]$_1$ [development strategy]$_2$ ...
- Lynx+MERS: According to the [development strategy]$_2$ [in the Chinese market]$_1$

The syntactic tree of the Chinese phrase "在 中国 市场 的 发展 战略" is shown in Figure 6. However, there are two TATs which can be applied to the source tree, as shown in Figure 7. The baseline system selects the left TAT and produces a monotone translation of the subtrees "$X_{\boxed{1}}$:PP" and "$X_{\boxed{2}}$:NPB". However, Lynx+MERS uses the right TAT and performs correct phrase reordering by swapping the two source phrases. Here the source tree is partially lexicalized, and both the contextual information and the information of sub-trees covered by nonterminals are considered by the MERS model.

6 Conclusion

In this paper, we propose a maximum entropy based rule selection model for syntax-based SMT. We use two kinds information as features: the local-contextual information of a rule, the information of sub-trees matched by nonterminals in a rule. During decoding, these features allow the decoder to perform a context-dependent rule selection. However, this information is never used in most of the current syntax-based SMT models.

The advantage of the MERS model is that it can help the decoder not only perform lexical selection, but also phrase reorderings. We demonstrate one way to incorporate the MERS models into a state-of-the-art linguistically syntax-based SMT model, the tree-to-string alignment model. Experiments show that by incorporating the MERS models, the baseline system achieves statistically significant improvements.

We find that rich contextual information can improve translation quality for a syntax-based SMT system. In future, we will explore more sophisticated features for the MERS model. Moreover, we will test the performance of the MERS model on large scale corpus.

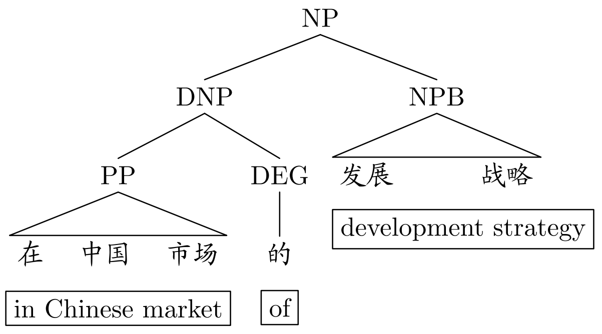


Figure 6: Syntactic tree of the source phrase "在 中国 市场 的 发展 战略".

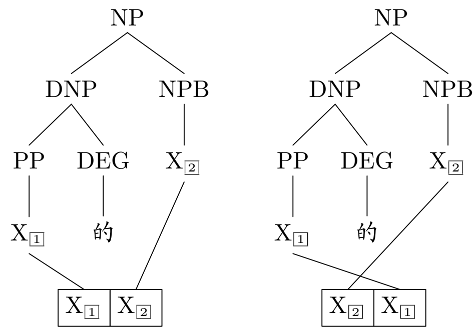


Figure 7: TATs which can be used for the source phrase "在 中国 市场 的 发展 战略".

Acknowledgements

We would like to thank Yajuan Lv for her valuable suggestions. This work was supported by the National Natural Science Foundation of China (NO. 60573188 and 60736014), and the High Technology Research and Development Program of China (NO. 2006AA010108).

References

Marine Carpuat and Dekai Wu. 2007a. How phrase sense disambiguation outperforms word sense disambiguation for statistical machine translation. In *11th Conference on Theoretical and Methodological Issues in Machine Translation*, pages 43–52.

Marine Carpuat and Dekai Wu. 2007b. Improving statistical machine translation using word sense disambiguation. In *Proceedings of EMNLP-CoNLL 2007*, pages 61–72.

Yee Seng Chan, Hwee Tou Ng, and David Chiang. 2007. Word sense disambiguation improves statistical machine translation. In *Proceedings of the 45th Annual*

Meeting of the Association for Computational Linguistics, pages 33–40.

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University Center for Research in Computing Technology.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 263–270.

M. Collins, P. Koehn, and I. Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proc. of ACL05*, pages 531–540.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of COLING-ACL 2006*, pages 961–968.

Zhongjun He, Qun Liu, and Shouxun Lin. 2008. Improving statistical machine translation using lexicalized rule selection. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 321–328.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of the 7th Biennial Conference of the Association for Machine Translation in the Americas*.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL 2003*, pages 127–133.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proceedings of the Sixth Conference of the Association for Machine Translation in the Americas*, pages 115–124.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*, pages 609–616.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 440–447.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Andreas Stolcke. 2002. Srilm – an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken language Processing*, volume 2, pages 901–904.

Deyi Xiong, Shuanglong Li, Qun Liu, Shouxun Lin, and Yueliang Qian. 2005. Parsing the penn chinese treebank with semantic knowledge. In *Proceedings of IJCNLP 2005*, pages 70–81.

Le Zhang. 2004. Maximum entropy modeling toolkit for python and c++. available at http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html.

Indirect-HMM-based Hypothesis Alignment for Combining Outputs from Machine Translation Systems

Xiaodong He[†], Mei Yang[‡] [*], Jianfeng Gao[†], Patrick Nguyen[†], and Robert Moore[†]

[†]Microsoft Research
One Microsoft Way
Redmond, WA 98052 USA
{xiaohe,jfgao, panguyen, bobmoore}@microsoft.com

[‡]Dept. of Electrical Engineering
University of Washington
Seattle, WA 98195, USA
yangmei@u.washington.edu

Abstract

This paper presents a new hypothesis alignment method for combining outputs of multiple machine translation (MT) systems. An indirect hidden Markov model (IHMM) is proposed to address the synonym matching and word ordering issues in hypothesis alignment. Unlike traditional HMMs whose parameters are trained via maximum likelihood estimation (MLE), the parameters of the IHMM are estimated *indirectly* from a variety of sources including word semantic similarity, word surface similarity, and a distance-based distortion penalty. The IHMM-based method significantly outperforms the state-of-the-art TER-based alignment model in our experiments on NIST benchmark datasets. Our combined SMT system using the proposed method achieved the best Chinese-to-English translation result in the constrained training track of the 2008 NIST Open MT Evaluation.

1 Introduction

System combination has been applied successfully to various machine translation tasks. Recently, confusion-network-based system combination algorithms have been developed to combine outputs of multiple machine translation (MT) systems to form a consensus output (Bangalore, et al. 2001, Matusov et al., 2006, Rosti et al., 2007, Sim et al., 2007). A confusion network comprises a sequence of sets of alternative words, possibly including *null*'s, with associated scores. The consensus output is then derived by selecting one word from each set of alternatives, to produce the sequence with the best overall score, which could be assigned in various ways such as by voting, by using posterior probability estimates, or by using a combination of these measures and other features.

Constructing a confusion network requires choosing one of the hypotheses as the backbone (also called "skeleton" in the literature), and other hypotheses are aligned to it at the word level. High quality hypothesis alignment is crucial to the performance of the resulting system combination. However, there are two challenging issues that make MT hypothesis alignment difficult. First, different hypotheses may use different synonymous words to express the same meaning, and these synonyms need to be aligned to each other. Second, correct translations may have different word orderings in different hypotheses and these words need to be properly reordered in hypothesis alignment.

In this paper, we propose an indirect hidden Markov model (IHMM) for MT hypothesis alignment. The HMM provides a way to model both synonym matching and word ordering. Unlike traditional HMMs whose parameters are trained via maximum likelihood estimation (MLE), the parameters of the IHMM are estimated *indirectly* from a variety of sources including word semantic similarity, word surface similarity, and a distance-based distortion penalty, without using large amount of training data. Our combined SMT system using the proposed method gave the best result on the Chinese-to-English test in the constrained training track of the 2008 NIST Open MT Evaluation (MT08).

2 Confusion-network-based MT system combination

The current state-of-the-art is confusion-network-based MT system combination as described by

[*] Mei Yang performed this work when she was an intern with Microsoft Research.

Rosti and colleagues (Rosti et al., 2007a, Rosti et al., 2007b). The major steps are illustrated in Figure 1. In Fig. 1 (a), hypotheses from different MT systems are first collected. Then in Fig. 1 (b), one of the hypotheses is selected as the backbone for hypothesis alignment. This is usually done by a sentence-level minimum Bayes risk (MBR) method which selects a hypothesis that has the minimum average distance compared to all hypotheses. The backbone determines the word order of the combined output. Then as illustrated in Fig. 1 (c), all other hypotheses are aligned to the backbone. Note that in Fig. 1 (c) the symbol ε denotes a *null* word, which is inserted by the alignment normalization algorithm described in section 3.4. Fig. 1 (c) also illustrates the handling of synonym alignment (e.g., aligning "car" to "sedan"), and word re-ordering of the hypothesis. Then in Fig. 1 (d), a confusion network is constructed based on the aligned hypotheses, which consists of a sequence of sets in which each word is aligned to a list of alternative words (including *null*) in the same set. Then, a set of global and local features are used to decode the confusion network.

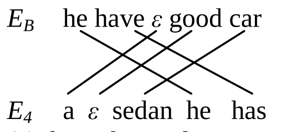
E_1	he have good car
E_2	he has nice sedan
E_3	it a nice car
E_4	a sedan he has

(a) hypothesis set

$$E_B = \arg\min_{E' \in \mathbf{E}} \sum_{E \in \mathbf{E}} TER(E', E)$$

e.g., $E_B = E_1$

(b) backbone selection


E_B he have ε good car
E_4 a ε sedan he has

(c) hypothesis alignment

he	have	ε	good	car
he	has	ε	nice	sedan
it	ε	a	nice	car
he	has	a	ε	sedan

(d) confusion network

Figure 1: Confusion-network-based MT system combination.

3 Indirect-HMM-based Hypothesis Alignment

In confusion-network-based system combination for SMT, a major difficulty is aligning hypotheses to the backbone. One possible statistical model for word alignment is the HMM, which has been widely used for bilingual word alignment (Vogel et al., 1996, Och and Ney, 2003). In this paper, we propose an indirect-HMM method for monolingual hypothesis alignment.

3.1 IHMM for hypothesis alignment

Let $e_1^I = (e_1, ..., e_I)$ denote the backbone, $e_1'^J = (e_1', ..., e_J')$ a hypothesis to be aligned to e_1^I, and $a_1^J = (a_1, ..., a_J)$ the alignment that specifies the position of the backbone word aligned to each hypothesis word. We treat each word in the backbone as an HMM state and the words in the hypothesis as the observation sequence. We use a first-order HMM, assuming that the emission probability $p(e_j' | e_{a_j})$ depends only on the backbone word, and the transition probability $p(a_j | a_{j-1}, I)$ depends only on the position of the last state and the length of the backbone. Treating the alignment as hidden variable, the conditional probability that the hypothesis is generated by the backbone is given by

$$p(e_1'^J | e_1^I) = \sum_{a_1^J} \prod_{j=1}^{J} \left[p(a_j | a_{j-1}, I) p(e_j' | e_{a_j}) \right] \quad (1)$$

As in HMM-based bilingual word alignment (Och and Ney, 2003), we also associate a *null* with each backbone word to allow generating hypothesis words that do not align to any backbone word.

In HMM-based hypothesis alignment, emission probabilities model the similarity between a backbone word and a hypothesis word, and will be referred to as the similarity model. The transition probabilities model word reordering, and will be called the distortion model.

3.2 Estimation of the similarity model

The similarity model, which specifies the emission probabilities of the HMM, models the similarity between a backbone word and a hypothesis word. Since both words are in the same language, the similarity model can be derived based on both semantic similarity and surface similarity, and the overall similarity model is a linear interpolation of the two:

$$p(e_j' | e_i) = \alpha \cdot p_{sem}(e_j' | e_i) + (1 - \alpha) \cdot p_{sur}(e_j' | e_i) \quad (2)$$

where $p_{sem}(e'_j \mid e_i)$ and $p_{sur}(e'_j \mid e_i)$ reflect the semantic and surface similarity between e'_j and e_i, respectively, and α is the interpolation factor.

Since the semantic similarity between two target words is source-dependent, the semantic similarity model is derived by using the source word sequence as a hidden layer:

$$\begin{aligned} & p_{sem}(e'_j \mid e_i) \\ & = \sum_{k=0}^{K} p(f_k \mid e_i) p(e'_j \mid f_k, e_i) \\ & \approx \sum_{k=0}^{K} p(f_k \mid e_i) p(e'_j \mid f_k) \end{aligned} \tag{3}$$

where $f_1^K = (f_1, ..., f_K)$ is the source sentence. Moreover, in order to handle the case that two target words are synonyms but neither of them has counter-part in the source sentence, a *null* is introduced on the source side, which is represented by f_0. The last step in (3) assumes that first e_i generates all source words including *null*. Then e_j' is generated by all source words including *null*.

In the common SMT scenario where a large amount of bilingual parallel data is available, we can estimate the translation probabilities from a source word to a target word and vice versa via conventional bilingual word alignment. Then both $p(f_k \mid e_i)$ and $p(e'_j \mid f_k)$ in (3) can be derived:

$$p(e'_j \mid f_k) = p_{s2t}(e'_j \mid f_k)$$

where $p_{s2t}(e'_j \mid f_k)$ is the translation model from the source-to-target word alignment model, and $p(f_k \mid e_i)$, which enforces the sum-to-1 constraint over all words in the source sentence, takes the following form,

$$p(f_k \mid e_i) = \frac{p_{t2s}(f_k \mid e_i)}{\sum_{k=0}^{K} p_{t2s}(f_k \mid e_i)}$$

where $p_{t2s}(f_k \mid e_i)$ is the translation model from the target-to-source word alignment model. In our method, $p_{t2s}(null \mid e_i)$ for all target words is simply a constant p_{null}, whose value is optimized on held-out data[1].

The surface similarity model can be estimated in several ways. A very simple model could be based on exact match: the surface similarity model, $p_{sur}(e'_j \mid e_i)$, would take the value 1.0 if $e' = e$, and 0 otherwise[2]. However, a smoothed surface similarity model is used in our method. If the target language uses alphabetic orthography, as English does, we treat words as letter sequences and the similarity measure can be the length of the longest matched prefix (LMP) or the length of the longest common subsequence (LCS) between them. Then, this raw similarity measure is transformed to a surface similarity score between 0 and 1 through an exponential mapping,

$$p_{sur}(e'_j \mid e_i) = \exp\left\{\rho \cdot \left[s(e'_j, e_i) - 1\right]\right\} \tag{4}$$

where $s(e'_j, e_i)$ is computed as

$$s(e'_j, e_i) = \frac{M(e'_j, e_i)}{\max(|e'_j|, |e_i|)}$$

and $M(e'_j, e_i)$ is the raw similarity measure of e_j' e_i, which is the length of the LMP or LCS of e_j' and e_i. and ρ is a smoothing factor that characterizes the mapping, Thus as ρ approaches infinity, $p_{sur}(e'_j \mid e_i)$ backs off to the exact match model. We found the smoothed similarity model of (4) yields slightly better results than the exact match model. Both LMP- and LCS- based methods achieve similar performance but the computation of LMP is faster. Therefore, we only report results of the LMP-based smoothed similarity model.

3.3 Estimation of the distortion model

The distortion model, which specifies the transition probabilities of the HMM, models the first-order dependencies of word ordering. In bilingual HMM-based word alignment, it is commonly assumed that transition probabilities

[1] The other direction, $p_{s2t}(e'_i \mid null)$, is available from the source-to-target translation model.

[2] Usually a small back-off value is assigned instead of 0.

$p(a_j = i \mid a_{j-1} = i', I)$ depend only on the jump distance $(i - i')$ (Vogel et al., 1996):

$$p(i \mid i', I) = \frac{c(i-i')}{\sum_{l=1}^{I} c(l-i')} \tag{5}$$

As suggested by Liang et al. (2006), we can group the distortion parameters $\{c(d)\}$, $d = i - i'$, into a few buckets. In our implementation, 11 buckets are used for $c(\leq -4)$, $c(-3)$, ... $c(0)$, ..., $c(5)$, $c(\geq 6)$. The probability mass for transitions with jump distance larger than 6 and less than -4 is uniformly divided. By doing this, only a handful of $c(d)$ parameters need to be estimated. Although it is possible to estimate them using the EM algorithm on a small development set, we found that a particularly simple model, described below, works surprisingly well in our experiments.

Since both the backbone and the hypothesis are in the same language, It seems intuitive that the distortion model should favor monotonic alignment and only allow non-monotonic alignment with a certain penalty. This leads us to use a distortion model of the following form, where K is a tuning factor optimized on held-out data.

$$c(d) = \left(1 + |d-1|\right)^{-K}, \; d = -4, \ldots, 6 \tag{6}$$

As shown in Fig. 2, the value of distortion score peaks at d=1, i.e., the monotonic alignment, and decays for non-monotonic alignments depending on how far it diverges from the monotonic alignment.

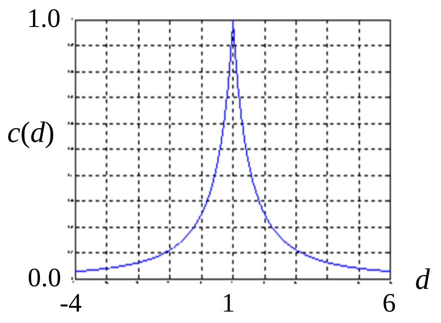


Figure 2, the distance-based distortion parameters computed according to (6), where K=2.

Following Och and Ney (2003), we use a fixed value p_0 for the probability of jumping to a *null* state, which can be optimized on held-out data, and the overall distortion model becomes

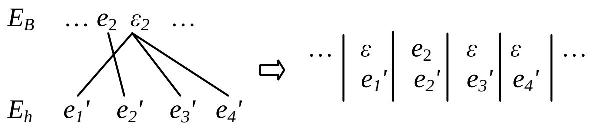
$$\tilde{p}(i \mid i', I) = \begin{cases} p_0 & \text{if } i = \textit{null} \text{ state} \\ (1-p_0) \cdot p(i \mid i', I) & \text{otherwise} \end{cases}$$

3.4 Alignment normalization

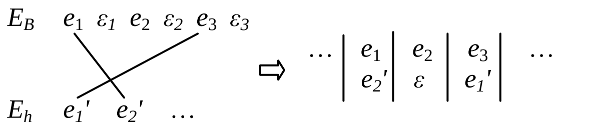
Given an HMM, the Viterbi alignment algorithm can be applied to find the best alignment between the backbone and the hypothesis,

$$\hat{a}_1^J = \underset{a_1^J}{\arg\max} \prod_{j=1}^{J} \left[p(a_j \mid a_{j-1}, I) \, p(e'_j \mid e_{a_j}) \right] \tag{7}$$

However, the alignment produced by the algorithm cannot be used directly to build a confusion network. There are two reasons for this. First, the alignment produced may contain 1-N mappings between the backbone and the hypothesis whereas 1-1 mappings are required in order to build a confusion network. Second, if hypothesis words are aligned to a *null* in the backbone or vice versa, we need to insert actual *null*s into the right places in the hypothesis and the backbone, respectively. Therefore, we need to normalize the alignment produced by Viterbi search.


E_B … e_2 ε_2 …
E_h e_1' e_2' e_3' e_4'
… | ε e_1' | e_2 e_2' | ε e_3' | ε e_4' | …

(a) hypothesis words are aligned to the backbone *null*


E_B e_1 ε_1 e_2 ε_2 e_3 ε_3
E_h e_1' e_2' …
… | e_1 e_2' | e_2 ε | e_3 e_1' | …

(b) a backbone word is aligned to no hypothesis word

Figure 3: illustration of alignment normalization

First, whenever more than one hypothesis words are aligned to one backbone word, we keep the link which gives the highest occupation probability computed via the forward-backward algorithm. The other hypothesis words originally aligned to the backbone word will be aligned to the *null* associated with that backbone word.

Second, for the hypothesis words that are aligned to a particular *null* on the backbone side, a set of *null*s are inserted around that backbone word associated with the *null* such that no links cross each other. As illustrated in Fig. 3 (a), if a hypothesis word e_2' is aligned to the backbone word e_2, a *null* is inserted in front of the backbone word e_2 linked to the hypothesis word e_1' that comes before e_2'. *Null*s are also inserted for other hypothesis words such as e_3' and e_4' after the backbone word e_2. If there is no hypothesis word aligned to that backbone word, all *null*s are inserted after that backbone word .[3]

For a backbone word that is aligned to no hypothesis word, a null is inserted on the hypothesis side, right after the hypothesis word which is aligned to the immediately preceding backbone word. An example is shown in Fig. 3 (b).

4 Related work

The two main hypothesis alignment methods for system combination in the previous literature are GIZA++ and TER-based methods. Matusov et al. (2006) proposed using GIZA++ to align words between different MT hypotheses, where all hypotheses of the test corpus are collected to create hypothesis pairs for GIZA++ training. This approach uses the conventional HMM model bootstrapped from IBM Model-1 as implemented in GIZA++, and heuristically combines results from aligning in both directions. System combination based on this approach gives an improvement over the best single system. However, the number of hypothesis pairs for training is limited by the size of the test corpus. Also, MT hypotheses from the same source sentence are correlated with each other and these hypothesis pairs are not i.i.d. data samples. Therefore, GIZA++ training on such a data set may be unreliable.

Bangalore et al. (2001) used a multiple string-matching algorithm based on Levenshtein edit distance, and later Sim et al. (2007) and Rosti et al. (2007) extended it to a TER-based method for hypothesis alignment. TER (Snover et al., 2006) measures the minimum number of edits, including substitution, insertion, deletion, and shift of blocks of words, that are needed to modify a hypothesis so that it exactly matches the other hypothesis. The best alignment is the one that gives the minimum number of translation edits. TER-based confusion network construction and system combination has demonstrated superior performance on various large-scale MT tasks (Rosti. et al, 2007). However, when searching for the optimal alignment, the TER-based method uses a strict surface hard match for counting edits. Therefore, it is not able to handle synonym matching well. Moreover, although TER-based alignment allows phrase shifts to accommodate the non-monotonic word ordering, all non-monotonic shifts are penalized equally no matter how short or how long the move is, and this penalty is set to be the same as that for substitution, deletion, and insertion edits. Therefore, its modeling of non-monotonic word ordering is very coarse-grained.

In contrast to the GIZA++-based method, our IHMM-based method has a similarity model estimated using bilingual word alignment HMMs that are trained on a large amount of bi-text data. Moreover, the surface similarity information is explicitly incorporated in our model, while it is only used implicitly via parameter initialization for IBM Model-1 training by Matusov et al. (2006). On the other hand, the TER-based alignment model is similar to a coarse-grained, non-normalized version of our IHMM, in which the similarity model assigns no penalty to an exact surface match and a fixed penalty to all substitutions, insertions, and deletions, and the distortion model simply assigns no penalty to a monotonic jump, and a fixed penalty to all other jumps, equal to the non-exact-match penalty in the similarity model.

There have been other hypothesis alignment methods. Karakos, et al. (2008) proposed an ITG-based method for hypothesis alignment, Rosti et al. (2008) proposed an incremental alignment method, and a heuristic-based matching algorithm was proposed by Jayaraman and Lavie (2005).

5 Evaluation

In this section, we evaluate our IHMM-based hypothesis alignment method on the Chinese-to-English (C2E) test in the constrained training track

[3] This only happens if no hypothesis word is aligned to a backbone word but some hypothesis words are aligned to the *null* associated with that backbone word.

of the 2008 NIST Open MT Evaluation (NIST, 2008). We compare to the TER-based method used by Rosti et al. (2007). In the following experiments, the NIST BLEU score is used as the evaluation metric (Papineni et al., 2002), which is reported as a percentage in the following sections.

5.1 Implementation details

In our implementation, the backbone is selected with MBR. Only the top hypothesis from each single system is considered as a backbone. A uniform posteriori probability is assigned to all hypotheses. TER is used as loss function in the MBR computation.

Similar to (Rosti et al., 2007), each word in the confusion network is associated with a word posterior probability. Given a system *S*, each of its hypotheses is assigned with a rank-based score of $1/(1+r)^{\eta}$, where *r* is the rank of the hypothesis, and η is a rank smoothing parameter. The system specific rank-based score of a word *w* for a given system *S* is the sum of all the rank-based scores of the hypotheses in system *S* that contain the word *w* at the given position (after hypothesis alignment). This score is then normalized by the sum of the scores of all the alternative words at the same position and from the same system *S* to generate the system specific word posterior. Then, the total word posterior of *w* over all systems is a sum of these system specific posteriors weighted by system weights.

Beside the word posteriors, we use language model scores and a word count as features for confusion network decoding.

Therefore, for an *M*-way system combination that uses *N* LMs, a total of M+N+1 decoding parameters, including M-1 system weights, one rank smoothing factor, *N* language model weights, and one weight for the word count feature, are optimized using Powell's method (Brent, 1973) to maximize BLEU score on a development set[4] .

Two language models are used in our experiments. One is a trigram model estimated from the English side of the parallel training data, and the other is a 5-gram model trained on the English GigaWord corpus from LDC using the MSRLM toolkit (Nguyen et al, 2007).

[4] The parameters of IHMM are not tuned by maximum-BLEU training.

In order to reduce the fluctuation of BLEU scores caused by the inconsistent translation output length, an unsupervised length adaptation method has been devised. We compute an expected length ratio between the MT output and the source sentences on the development set after maximum-BLEU training. Then during test, we adapt the length of the translation output by adjusting the weight of the word count feature such that the expected output/source length ratio is met. In our experiments, we apply length adaptation to the system combination output at the level of the whole test corpus.

5.2 Development and test data

The development (dev) set used for system combination parameter training contains 1002 sentences sampled from the previous NIST MT Chinese-to-English test sets: 35% from MT04, 55% from MT05, and 10% from MT06-newswire. The test set is the MT08 Chinese-to-English "current" test set, which includes 1357 sentences from both newswire and web-data genres. Both dev and test sets have four references per sentence.

As inputs to the system combination, 10-best hypotheses for each source sentence in the dev and test sets are collected from each of the eight single systems. All outputs on the MT08 test set were true-cased before scoring using a log-linear conditional Markov model proposed by Toutanova et al. (2008). However, to save computation effort, the results on the dev set are reported in case insensitive BLEU (ciBLEU) score instead.

5.3 Experimental results

In our main experiments, outputs from a total of eight single MT systems were combined. As listed in Table 1, Sys-1 is a tree-to-string system proposed by Quirk et al., (2005); Sys-2 is a phrase-based system with fast pruning proposed by Moore and Quirk (2008); Sys-3 is a phrase-based system with syntactic source reordering proposed by Wang et al. (2007a); Sys-4 is a syntax-based pre-ordering system proposed by Li et. al. (2007); Sys-5 is a hierarchical system proposed by Chiang (2007); Sys-6 is a lexicalized re-ordering system proposed by Xiong et al. (2006); Sys-7 is a two-pass phrase-based system with adapted LM proposed by Foster and Kuhn (2007); and Sys-8 is a hierarchical system with two-pass rescoring using a parser-based LM proposed by Wang et al., (2007b). All systems were trained within the confines of the constrained training condition of NIST MT08 evaluation. These single systems are optimized with maximum-BLEU training on different subsets of the previous NIST MT test data. The bilingual translation models used to compute the semantic similarity are from the word-dependent HMMs proposed by He (2007), which are trained on two million parallel sentence-pairs selected from the training corpus allowed by the constrained training condition of MT08.

5.3.1 Comparison with TER alignment

In the IHMM-based method, the smoothing factor for surface similarity model is set to $\rho = 3$, the interpolation factor of the overall similarity model is set to $\alpha = 0.3$, and the controlling factor of the distance-based distortion parameters is set to K=2. These settings are optimized on the dev set. Individual system results and system combination results using both IHMM and TER alignment, on both the dev and test sets, are presented in Table 1. The TER-based hypothesis alignment tool used in our experiments is the publicly available TER Java program, TERCOM (Snover et al., 2006). Default settings of TERCOM are used in the following experiments.

On the dev set, the case insensitive BLEU score of the IHMM-based 8-way system combination output is about 5.8 points higher than that of the best single system. Compared to the TER-based method, the IHMM-based method is about 1.5 BLEU points better. On the MT08 test set, the IHMM-based system combination gave a case sensitive BLEU score of 30.89%. It outperformed the best single system by 4.7 BLEU points and the TER-based system combination by 1.0 BLEU points. Note that the best single system on the dev set and the test set are different. The different single systems are optimized on different tuning sets, so this discrepancy between dev set and test set results is presumably due to differing degrees of mismatch between the dev and test sets and the various tuning sets.

Table 1. Results of single and combined systems on the dev set and the MT08 test set

System	Dev ciBLEU%	MT08 BLEU%
System 1	34.08	21.75
System 2	33.78	20.42
System 3	34.75	21.69
System 4	37.85	25.52
System 5	37.80	24.57
System 6	37.28	24.40
System 7	32.37	25.51
System 8	34.98	26.24
TER	42.11	29.89
IHMM	43.62	30.89

In order to evaluate how well our method performs when we combine more systems, we collected MT outputs on MT08 from seven additional single systems as summarized in Table 2. These systems belong to two groups. Sys-9 to Sys-12 are in the first group. They are syntax-augmented hierarchical systems similar to those described by Shen et al. (2008) using different Chinese word segmentation and language models. The second group has Sys-13 to Sys-15. Sys-13 is a phrasal system proposed by Koehn et al. (2003), Sys-14 is a hierarchical system proposed by Chiang (2007), and Sys-15 is a syntax-based system proposed by Galley et al. (2006). All seven systems were trained within the confines of the constrained training condition of NIST MT08 evaluation.

We collected 10-best MT outputs only on the MT08 test set from these seven extra systems. No MT outputs on our dev set are available from them at present. Therefore, we directly adopt system combination parameters trained for the previous 8-way system combination, except the system weights, which are re-set by the following heuristics: First, the total system weight mass 1.0 is evenly divided among the three groups of single systems: {Sys-1~8}, {Sys-9~12}, and {Sys-13~15}. Each group receives a total system weight mass of 1/3. Then the weight mass is further divided in each group: in the first group, the original weights of systems 1~8 are multiplied by 1/3; in the second and third groups, the weight mass is evenly distributed within the group, i.e., 1/12 for each system in group 2, and 1/9 for each system in group 3[5]. Length adaptation is applied to control the final output length, where the same expected length ratio of the previous 8-way system combination is adopted.

The results of the 15-way system combination are presented in Table 3. It shows that the IHMM-based method is still about 1 BLEU point better than the TER-based method. Moreover, combining 15 single systems gives an output that has a NIST BLEU score of 34.82%, which is 3.9 points better than the best submission to the NIST MT08 constrained training track (NIST, 2008). To our knowledge, this is the best result reported on this task.

Table 2. Results of seven additional single systems on the NIST MT08 test set

System	MT08 BLEU%
System 9	29.59
System 10	29.57
System 11	29.64
System 12	29.85
System 13	25.53
System 14	26.04
System 15	29.70

Table 3. Results of the 15-way system combination on the NIST MT08 C2E test set

Sys. Comb.	MT08 BLEU%
TER	33.81
IHMM	34.82

5.3.2 Effect of the similarity model

In this section, we evaluate the effect of the semantic similarity model and the surface similarity model by varying the interpolation weight α of (2). The results on both the dev and test sets are reported in Table 4. In one extreme case, $\alpha = 1$, the overall similarity model is based only on semantic similarity. This gives a case insensitive BLEU score of 41.70% and a case sensitive BLEU score of 28.92% on the dev and test set, respectively. The accuracy is significantly improved to 43.62% on the dev set and 30.89% on test set when $\alpha = 0.3$. In another extreme case, $\alpha = 0$, in which only the surface similarity model is used for the overall similarity model, the performance degrades by about 0.2 point. Therefore, the surface similarity information seems more important for monolingual hypothesis alignment, but both sub-models are useful.

[5] This is just a rough guess because no dev set is available. We believe a better set of system weights could be obtained if MT outputs on a common dev set were available.

Table 4. Effect of the similarity model

	Dev ciBLEU%	Test BLEU%
$\alpha = 1.0$	41.70	28.92
$\alpha = 0.7$	42.86	30.50
$\alpha = 0.5$	43.11	30.94
$\alpha = 0.3$	43.62	30.89
$\alpha = 0.0$	43.35	30.73

5.3.3 Effect of the distortion model

We investigate the effect of the distance-based distortion model by varying the controlling factor K in (6). For example, setting K=1.0 gives a linear-decay distortion model, and setting K=2.0 gives a quadratic smoothed distance-based distortion model. As shown in Table 5, the optimal result can be achieved using a properly smoothed distance-based distortion model.

Table 5. Effect of the distortion model

	Dev ciBLEU%	Test BLEU%
K=1.0	42.94	30.44
K=2.0	43.62	30.89
K=4.0	43.17	30.30
K=8.0	43.09	30.01

6 Conclusion

Synonym matching and word ordering are two central issues for hypothesis alignment in confusion-network-based MT system combination. In this paper, an IHMM-based method is proposed for hypothesis alignment. It uses a similarity model for synonym matching and a distortion model for word ordering. In contrast to previous methods, the similarity model explicitly incorporates both semantic and surface word similarity, which is critical to monolingual word alignment, and a smoothed distance-based distortion model is used to model the first-order dependency of word ordering, which is shown to be better than simpler approaches.

Our experimental results show that the IHMM-based hypothesis alignment method gave superior results on the NIST MT08 C2E test set compared to the TER-based method. Moreover, we show that our system combination method can scale up to combining more systems and produce a better output that has a case sensitive BLEU score of 34.82, which is 3.9 BLEU points better than the best official submission of MT08.

Acknowledgement

The authors are grateful to Chris Quirk, Arul Menezes, Kristina Toutanova, William Dolan, Mu Li, Chi-Ho Li, Dongdong Zhang, Long Jiang, Ming Zhou, George Foster, Roland Kuhn, Jing Zheng, Wen Wang, Necip Fazil Ayan, Dimitra Vergyri, Nicolas Scheffer, Andreas Stolcke, Kevin Knight, Jens-Soenke Voeckler, Spyros Matsoukas, and Antti-Veikko Rosti for assistance with the MT systems and/or for the valuable suggestions and discussions.

References

Srinivas Bangalore, German Bordel, and Giuseppe Riccardi. 2001. Computing consensus translation from multiple machine translation systems. In *Proc. of IEEE ASRU*, pp. 351–354.

Richard Brent, 1973. Algorithms for Minimization without Derivatives. Prentice-Hall, Chapter 7.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

George Foster and Roland Kuhn. 2007. Mixture-Model Adaptation for SMT. In *Proc. of the Second ACL Workshop on Statistical Machine Translation*. pp. 128 – 136.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang and Ignacio Thayer. 2006. Scalable Inference and Training of Context-Rich Syntactic Translation Models. In *Proc. of COLING-ACL*, pp. 961–968.

Xiaodong He. 2007. Using Word-Dependent Transition Models in HMM based Word Alignment for Statistical Machine Translation. In *Proc. of the Second ACL Workshop on Statistical Machine Translation*.

Shyamsundar Jayaraman and Alon Lavie. 2005. Multi-engine machine translation guided by explicit word matching. In *Proc. of EAMT*. pp. 143 – 152.

Damianos Karakos, Jason Eisner, Sanjeev Khudanpur, and Markus Dreyer. 2008. Machine Translation System Combination using ITG-based Alignments. In *Proc. of ACL-HLT*, pp. 81–84.

Chi-Ho Li, Minghui Li, Dongdong Zhang, Mu Li, Ming Zhou, Yi Guan. 2007. A Probabilistic Approach to Syntax-based Reordering for Statistical Machine Translation. In *Proc. of ACL*. pp. 720 – 727.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by Agreement. In *Proc. of NAACL*. pp 104 – 111.

Evgeny Matusov, Nicola Ueffing, and Hermann Ney. 2006. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. In *Proc. of EACL*, pp. 33–40.

Robert Moore and Chris Quirk. 2007. Faster Beam-Search Decoding for Phrasal Statistical Machine Translation. In *Proc. of MT Summit XI*.

Patrick Nguyen, Jianfeng Gao and Milind Mahajan. 2007. MSRLM: a scalable language modeling toolkit. *Microsoft Research Technical Report MSR-TR-2007-144.*

NIST. 2008. The 2008 NIST Open Machine Translation Evaluation. www.nist.gov/speech/tests/mt/2008/doc/

Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL*, pp. 311–318.

Koehn, Philipp, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase based translation. In *Proc. of NAACL*. pp. 48 – 54.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proc. of ACL*. pp. 271–279.

Antti-Veikko I. Rosti, Bing Xiang, Spyros Matsoukas, Richard Schwartz, Necip Fazil Ayan, and Bonnie J. Dorr. 2007a. Combining outputs from multiple machine translation systems. In *Proc. of NAACL-HLT*, pp. 228–235.

Antti-Veikko I. Rosti, Spyros Matsoukas, and Richard Schwartz. 2007b. Improved Word-Level System Combination for Machine Translation. In *Proc. of ACL*, pp. 312–319.

Antti-Veikko I. Rosti, Bing Zhang, Spyros Matsoukas, and Richard Schwartz. 2008. Incremental Hypothesis Alignment for Building Confusion Networks with Application to Machine Translation System Combination, In *Proc. of the Third ACL Workshop on Statistical Machine Translation*, pp. 183–186.

Libin Shen, Jinxi Xu, Ralph Weischedel. 2008. A New String-to-Dependency Machine Translation Algorithm with a Target Dependency Language Model. In *Proc. of ACL-HLT*, pp. 577–585.

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. of ICASSP*, *vol. 4*. pp. 105–108.

Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proc. of AMTA*.

Kristina Toutanova, Hisami Suzuki and Achim Ruopp. 2008. Applying Morphology Generation Models to Machine Translation. In *Proc. of ACL*. pp. 514 – 522.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based Word Alignment In Statistical Translation. In *Proc. of COLING*. pp. 836-841.

Chao Wang, Michael Collins, and Philipp Koehn. 2007a. Chinese Syntactic Reordering for Statistical Machine Translation. In *Proc. of EMNLP-CoNLL*. pp. 737-745.

Wen Wang, Andreas Stolcke, Jing Zheng. 2007b. Reranking Machine Translation Hypotheses With Structured and Web-based Language Models. In *Proc. of IEEE ASRU*. pp. 159 – 164.

Deyi Xiong, Qun Liu and Shouxun Lin. 2006. Maximum Entropy Based Phrase Reordering Model for Statistical Machine Translation. In *Proc. of ACL*. pp. 521 – 528.

Coarse-to-Fine Syntactic Machine Translation using Language Projections

Slav Petrov **Aria Haghighi** **Dan Klein**
Computer Science Division, EECS Department
University of California at Berkeley
Berkeley, CA 94720
{petrov, aria42, klein}@eecs.berkeley.edu

Abstract

The intersection of tree transducer-based translation models with n-gram language models results in huge dynamic programs for machine translation decoding. We propose a multipass, coarse-to-fine approach in which the language model complexity is incrementally introduced. In contrast to previous *order-based* bigram-to-trigram approaches, we focus on *encoding-based* methods, which use a clustered encoding of the target language. Across various encoding schemes, and for multiple language pairs, we show speed-ups of up to 50 times over single-pass decoding while improving BLEU score. Moreover, our entire decoding cascade for trigram language models is faster than the corresponding bigram pass alone of a bigram-to-trigram decoder.

1 Introduction

In the absence of an n-gram language model, decoding a synchronous CFG translation model is very efficient, requiring only a variant of the CKY algorithm. As in monolingual parsing, dynamic programming items are simply indexed by a source language span and a syntactic label. Complexity arises when n-gram language model scoring is added, because items must now be distinguished by their initial and final few target language words for purposes of later combination. This lexically exploded search space is a root cause of inefficiency in decoding, and several methods have been suggested to combat it. The approach most relevant to the current work is Zhang and Gildea (2008), which begins with an initial bigram pass and uses the resulting chart to guide a final trigram pass. Substantial speed-ups are obtained, but computation is still dominated by the initial bigram pass. The key challenge is that unigram models are too poor to prune well, but bigram models are already huge. In short, the problem is that there are too many words in the target language. In this paper, we propose a new, coarse-to-fine, multipass approach which allows much greater speedups by translating into *abstracted languages*. That is, rather than beginning with a *low-order* model of a still-large language, we exploit *language projections*, hierarchical clusterings of the target language, to effectively reduce the size of the target language. In this way, initial passes can be very quick, with complexity phased in gradually.

Central to coarse-to-fine language projection is the construction of sequences of word clusterings (see Figure 1). The clusterings are deterministic mappings from words to clusters, with the property that each clustering refines the previous one. There are many choice points in this process, including how these clusterings are obtained and how much refinement is optimal for each pass. We demonstrate that likelihood-based hierarchical EM training (Petrov et al., 2006) and cluster-based language modeling methods (Goodman, 2001) are superior to both rank-based and random-projection methods. In addition, we demonstrate that more than two passes are beneficial and show that our computation is equally distributed over all passes. In our experiments, passes with less than 16-cluster language models are most advantageous, and even a single pass with just two word clusters can reduce decoding time greatly.

To follow related work and to focus on the effects of the language model, we present translation results under an inversion transduction grammar (ITG) translation model (Wu, 1997) trained on the Europarl corpus (Koehn, 2005), described in detail in Section 3, and using a trigram language model. We show that, on a range of languages, our coarse-to-fine decoding approach greatly outperforms baseline beam pruning and bigram-to-trigram pruning on time-to-BLEU plots, reducing decoding times by up to a factor of 50 compared to single pass decoding. In addition, coarse-to-fine decoding increases BLEU scores by up to 0.4 points. This increase is a mixture of improved search and subtly advantageous coarse-to-fine effects which are further discussed below.

2 Coarse-to-Fine Decoding

In coarse-to-fine decoding, we create a series of initially simple but increasingly complex search problems. We then use the solutions of the simpler problems to prune the search spaces for more complex models, reducing the total computational cost.

2.1 Related Work

Taken broadly, the coarse-to-fine approach is not new to machine translation (MT) or even syntactic MT. Many common decoder precomputations can be seen as coarse-to-fine methods, including the A*-like forward estimates used in the Moses decoder (Koehn et al., 2007). In an ITG framework like ours, Zhang and Gildea (2008) consider an approach in which the results of a bigram pass are used as an A* heuristic to guide a trigram pass. In their two-pass approach, the coarse bigram pass becomes computationally dominant. Our work differs in two ways. First, we use posterior pruning rather than A* search. Unlike A* search, posterior pruning allows multipass methods. Not only are posterior pruning methods simpler (for example, there is no need to have complex multipart bounds), but they can be much more effective. For example, in monolingual parsing, posterior pruning methods (Goodman, 1997; Charniak et al., 2006; Petrov and Klein, 2007) have led to greater speedups than their more cautious A* analogues (Klein and Manning, 2003; Haghighi et al., 2007), though at the cost of guaranteed optimality.

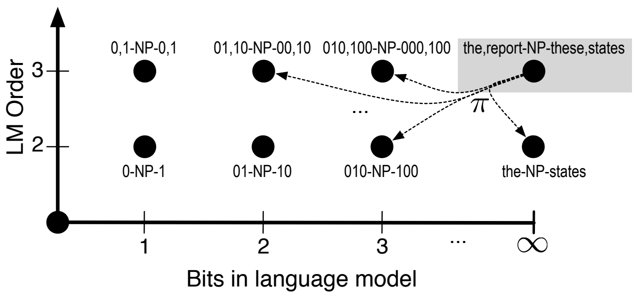


Figure 2: Possible state projections π for the target noun phrase *"the report for these states"* using the clusters from Figure 1. The number of bits used to encode the target language vocabulary is varied along the x-axis. The language model order is varied along the y-axis.

Second, we focus on an orthogonal axis of abstraction: the size of the target language. The introduction of abstract languages gives better control over the granularity of the search space and provides a richer set of intermediate problems, allowing us to adapt the level of refinement of the intermediate, coarse passes to minimize total computation.

Beyond coarse-to-fine approaches, other related approaches have also been demonstrated for syntactic MT. For example, Venugopal et al. (2007) considers a greedy first pass with a full model followed by a second pass which bounds search to a region near the greedy results. Huang and Chiang (2007) searches with the full model, but makes assumptions about the the amount of reordering the language model can trigger in order to limit exploration.

2.2 Language Model Projections

When decoding in a syntactic translation model with an n-gram language model, search states are specified by a grammar nonterminal X as well as the the n-1 left-most target side words $l_{n-1}, \ldots, l_1$ and right-most target side words $r_1, \ldots, r_{n-1}$ of the generated hypothesis. We denote the resulting lexicalized state as $l_{n-1}, \ldots, l_1$-X-$r_1, \ldots, r_{n-1}$. Assuming a vocabulary V and grammar symbol set G, the state space size is up to $|V|^{2(n-1)}|G|$, which is immense for a large vocabulary when $n > 1$. We consider two ways to reduce the size of this search space. First, we can reduce the order of the language model. Second, we can reduce the number of words in the vocabulary. Both can be thought of as *projections* of the search space to smaller ab-

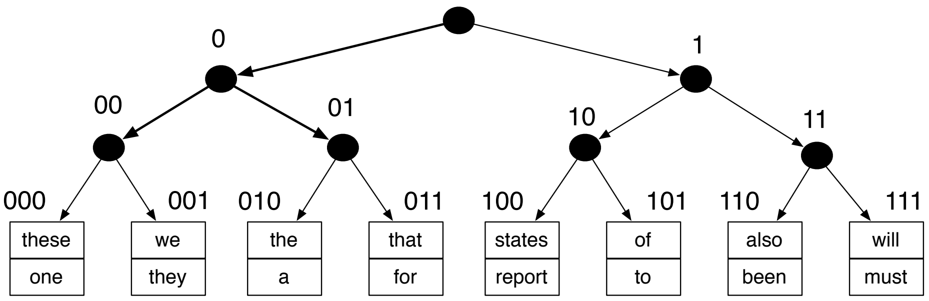


Figure 1: An example of hierarchical clustering of target language vocabulary (see Section 4). Even with a small number of clusters our divisive HMM clustering (Section 4.3) captures sensible syntactico-semantic classes.

stracted spaces. Figure 2 illustrates those two orthogonal axes of abstraction.

Order-based projections are simple. As shown in Figure 2, they simply strip off the appropriate words from each state, collapsing dynamic programming items which are identical from the standpoint of their left-to-right combination in the lower order language model. However, having only order-based projections is very limiting. Zhang and Gildea (2008) found that their computation was dominated by their bigram pass. The only lower-order pass possible uses a unigram model, which provides no information about the interaction of the language model and translation model reorderings. We therefore propose *encoding-based* projections. These projections reduce the size of the target language vocabulary by deterministically projecting each target language word to a word cluster. This projection extends to the whole search state in the obvious way: assuming a bigram language model, the state l-X-r projects to $c(l)$-X-$c(r)$, where $c(\cdot)$ is the deterministic word-to-cluster mapping.

In our multipass approach, we will want a sequence $c_1 \dots c_n$ of such projections. This requires a *hierarchical clustering* of the target words, as shown in Figure 1. Each word's cluster membership can be represented by an n-bit binary string. Each prefix of length k declares that word's cluster assignment at the k-bit level. As we vary k, we obtain a sequence of projections $c_k(\cdot)$, each one mapping words to a more refined clustering. When performing inference in a k-bit projection, we replace the detailed original language model over words with a coarse language model LM$_k$ over the k-bit word clusters. In addition, we replace the phrase table with a projected phrase table, which further increases the speed of projected passes. In Section 4, we describe the various clustering schemes explored, as well as how the coarse LM$_k$ are estimated.

2.3 Multipass Decoding

Unlike previous work, where the state space exists only at two levels of abstraction (i.e. bigram and trigram), we have multiple levels to choose from (Figure 2). Because we use both encoding-based and order-based projections, our options form a lattice of coarser state spaces, varying from extremely simple (a bigram model with just two word clusters) to nearly the full space (a trigram model with 10 bits or 1024 word clusters).

We use this lattice to perform a series of coarse passes with increasing complexity. More formally, we decode a source sentence multiple times, in a sequence of state spaces $S_0, S_1, \dots, S_n{=}S$, where each S_i is a refinement of S_{i-1} in either language model order, language encoding size, or both. The state spaces S_i and S_j ($i < j$) are related to each other via a projection operator $\pi_{j \to i}(\cdot)$ which maps refined states deterministically to coarser states.

We start by decoding an input x in the simplest state space S_0. In particular, we compute the chart of the posterior distributions $p_0(s) = P(s|x)$ for all states $s \in S_0$. These posteriors will be used to prune the search space S_1 of the following pass. States s whose posterior falls below a threshold t trigger the removal of all more refined states s' in the subsequent pass (see Figure 3). This technique is *posterior pruning*, and is different from A* methods in two main ways. First, it can be iterated in a multipass setting, and, second, it is generally more effi-

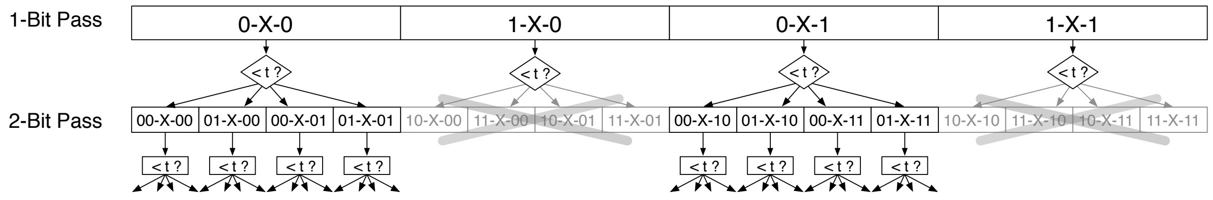


Figure 3: Example of state pruning in coarse-to-fine decoding using the language encoding projection (see Section 2.2). During the coarse one-bit word cluster pass, two of the four possible states are pruned. Every extension of the pruned one-bit states (indicated by the grey shading) are not explored during the two-bit word cluster pass.

cient with a potential cost of increased search errors (see Section 2.1 for more discussion).

Looking at Figure 2, multipass coarse-to-fine decoding can be visualized as a walk from a coarse point somewhere in the lower left to the most refined point in the upper right of the grid. Many coarse-to-fine schedules are possible. In practice, we might start decoding with a 1-bit word bigram pass, followed by an 3-bit word bigram pass, followed by a 5-bit word trigram pass and so on (see Section 5.3 for an empirical investigation). In terms if time, we show that coarse-to-fine gives substantial speed-ups. There is of course an additional memory requirement, but it is negligible. As we will see in our experiments (Section 5) the largest gains can be obtained with extremely coarse language models. In particular, the largest coarse model we use in our best multipass decoder uses a 4-bit encoding and hence has only 16 distinct words (or at most 4096 trigrams).

3 Inversion Transduction Grammars

While our approach applies in principle to a variety of machine translation systems (phrase-based or syntactic), we will use the inversion transduction grammar (ITG) approach of Wu (1997) to facilitate comparison with previous work (Zens and Ney, 2003; Zhang and Gildea, 2008) as well as to focus on language model complexity. ITGs are a subclass of synchronous context-free grammars (SCFGs) where there are only three kinds of rules. Preterminal unary productions produce terminal strings on both sides (words or phrases): $X \rightarrow \mathbf{e}/\mathbf{f}$. Binary in-order productions combine two phrases monotonically ($X \rightarrow [YZ]$). Finally, binary inverted productions invert the order of their children ($X \rightarrow \langle YZ \rangle$). These productions are associated with rewrite weights in the standard way.

Without a language model, SCFG decoding is just like (monolingual) CFG parsing. The dynamic programming states are specified by ${}_iX_j$, where $\langle i, j \rangle$ is a source sentence span and X is a nonterminal. The only difference is that whenever we apply a CFG production on the source side, we need to remember the corresponding synchronous production on the target side and store the best obtainable translation via a backpointer. See Wu (1996) or Melamed (2004) for a detailed exposition.

Once we integrate an n-gram language model, the state space becomes lexicalized and combining dynamic programming items becomes more difficult. Each state is now parametrized by the initial and final $n-1$ words in the target language hypothesis: $l_{n-1}, ..., l_1\text{-}{}_iX_j\text{-}r_1, ..., r_{n-1}$. Whenever we combine two dynamic programming items, we need to score the fluency of their concatentation by incorporating the score of any language model features which cross the target side boundaries of the two concatenated items (Chiang, 2005). Decoding with an integrated language model is computationally expensive for two reasons: (1) the need to keep track of a large number of lexicalized hypotheses for each source span, and (2) the need to frequently query the large language model for each hypothesis combination.

Multipass coarse-to-fine decoding can alleviate both computational issues. We start by decoding in an extremely coarse bigram search space, where there are very few possible translations. We compute standard inside/outside probabilities (iS/oS), as follows. Consider the application of non-inverted binary rule: we combine two items $l_b\text{-}{}_iB_k\text{-}r_b$ and $l_c\text{-}{}_kC_j\text{-}r_c$ spanning $\langle i, k \rangle$ and $\langle k, j \rangle$ respectively to form a larger item $l_b\text{-}{}_iA_j\text{-}r_c$, spanning $\langle i, j \rangle$. The

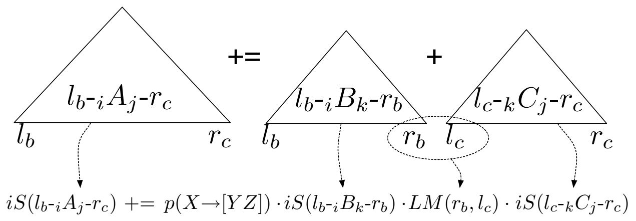

l_b-$_i A_j$-r_c
+=
l_b-$_i B_k$-r_b
+
l_c-$_k C_j$-r_c
l_b
r_c
l_b
r_b
l_c
r_c
$iS(l_b\text{-}_iA_j\text{-}r_c)$ += $p(X\rightarrow[YZ]) \cdot iS(l_b\text{-}_iB_k\text{-}r_b) \cdot LM(r_b,l_c) \cdot iS(l_c\text{-}_kC_j\text{-}r_c)$

Figure 4: Monotonic combination of two hypotheses during the inside pass involves scoring the fluency of the concatenation with the language model.

inside score of the new item is incremented by:

$$iS(l_b\text{-}_iA_j\text{-}r_c) \mathrel{+}= p(X \rightarrow [YZ]) \cdot iS(l_b\text{-}_iB_k\text{-}r_b) \cdot iS(l_c\text{-}_kC_j\text{-}r_c) \cdot LM(r_b, l_c)$$

This process is also illustrated in Figure 4. Of course, we also loop over the split point k and apply the other two rule types (inverted concatenation, terminal generation). We omit those cases from this exposition, as well as the update for the outside pass; they are standard and similar. Once we have computed the inside and outside scores, we compute posterior probabilities for all items:

$$p(l_a\text{-}_iA_j\text{-}r_a) = \frac{iS(l_a\text{-}_iA_j\text{-}r_a)oS(l_a\text{-}_iA_j\text{-}r_a)}{iS(root)}$$

where $iS(root)$ is sum of all translations' scores. States with low posteriors are then pruned away. We proceed to compute inside/outside score in the next, more refined search space, using the projections $\pi_{i\rightarrow i-1}$ to map between states in S_i and S_{i-1}. In each pass, we skip all items whose projection into the previous stage had a probability below a stage-specific threshold. This process is illustrated in Figure 3. When we reach the most refined search space S_∞, we do not prune, but rather extract the Viterbi derivation instead.[1]

4 Learning Coarse Languages

Central to our encoding-based projections (see Section 2.2) are hierarchical clusterings of the target language vocabulary. In the present work, these clusterings are each k-bit encodings and yield sequences of coarse language models LM$_k$ and phrasetables PT$_k$.

[1]Other final decoding strategies are possible, of course, including variational methods and minimum-risk methods (Zhang and Gildea, 2008).

Given a hierarchical clustering, we estimate the corresponding LM$_k$ from a corpus obtained by replacing each token in a target language corpus with the appropriate word cluster. As with our original refined language model, we estimate each coarse language model using the SRILM toolkit (Stolcke, 2002). The phrasetables PT$_k$ are similarly estimated by replacing the words on the target side of each phrase pair with the corresponding cluster. This procedure can potentially map two distinct phrase pairs to the same coarse translation. In such cases we keep only one coarse phrase pair and sum the scores of the colliding originals.

There are many possible schemes for creating hierarchical clusterings. Here, we consider several divisive clustering methods, where coarse word clusters are recursively split into smaller subclusters.

4.1 Random projections

The simplest approach to splitting a cluster is to randomly assign each word type to one of two new subclusters. Random projections have been shown to be a good and computationally inexpensive dimensionality reduction technique, especially for high dimensional data (Bingham and Mannila, 2001). Although our best performance does not come from random projections, we still obtain substantial speed-ups over a single pass fine decoder when using random projections in coarse passes.

4.2 Frequency clustering

In frequency clustering, we allocate words to clusters by frequency. At each level, the most frequent words go into one cluster and the rarest words go into another one. Concretely, we sort the words in a given cluster by frequency and split the cluster so that the two halves have equal token mass. This approach can be seen as a radically simplified version of Brown et al. (1992). It can, and does, result in highly imbalanced cluster hierarchies.

4.3 HMM clustering

An approach found to be effective by Petrov and Klein (2007) for coarse-to-fine parsing is to use likelihood-based hierarchical EM training. We adopt this approach here by identifying each cluster with a latent state in an HMM and determinizing the emissions so that each word type is emitted

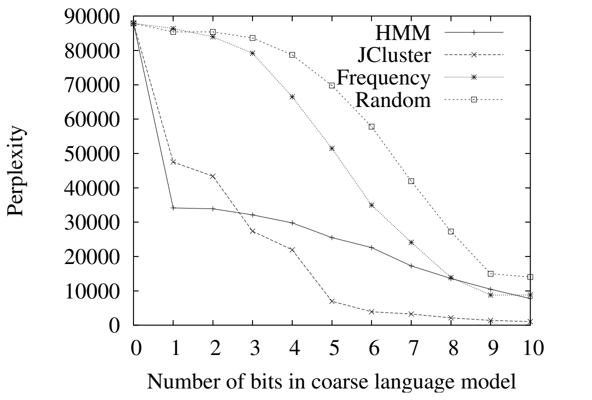


Figure 5: Results of coarse language model perplexity experiment (see Section 4.5). HMM and JClustering have lower perplexity than frequency and random clustering for all number of bits in the language encoding.

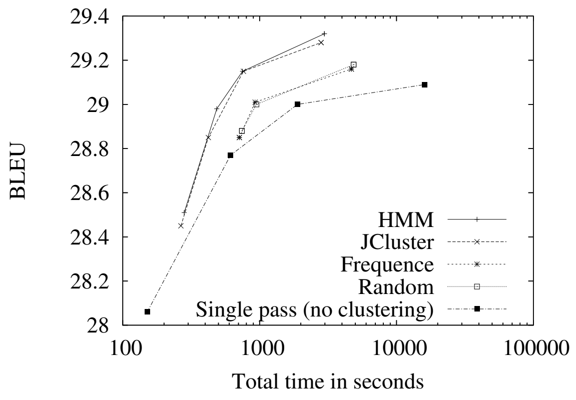


Figure 6: Coarse-to-fine decoding with HMM or JClustering coarse language models reduce decoding times while increasing accuracy.

by only one state. When splitting a cluster s into s_1 and s_2, we initially clone and mildly perturb its corresponding state. We then use EM to learn parameters, which splits the state, and determinize the result. Specifically, each word w is assigned to s_1 if $P(w|s_1) > P(w|s_2)$ and s_2 otherwise. Because of this determinization after each round of EM, a word in one cluster will be allocated to exactly one of that cluster's children. This process not only guarantees that the clusters are hierarchical, it also avoids the state drift discussed by Petrov and Klein (2007). Because the emissions are sparse, learning is very efficient. An example of some of the words associated with early splits can be seen in Figure 1.

4.4 JCluster

Goodman (2001) presents a clustering scheme which aims to minimize the entropy of a word given a cluster. This is accomplished by incrementally swapping words between clusters to locally minimize entropy.[2] This clustering algorithm was developed with a slightly different application in mind, but fits very well into our framework, because the hierarchical clusters it produces are trained to maximize predictive likelihood.

4.5 Clustering Results

We applied the above clustering algorithms to our monolingual language model data to obtain hierarchical clusters. We then trained coarse language models of varying granularity and evaluated them on a held-out set. To measure the quality of the coarse language models we use perplexity (exponentiated cross-entropy).[3] Figure 5 shows that HMM clustering and JClustering have lower perplexity than frequency and random based clustering for all complexities. In the next section we will present a set of machine translation experiments using these coarse language models; the clusterings with better perplexities generally produce better decoders.

5 Experiments

We ran our experiments on the Europarl corpus (Koehn, 2005) and show results on Spanish, French and German to English translation. We used the setup and preprocessing steps detailed in the 2008 Workshop on Statistical Machine Translation.[4] Our baseline decoder uses an ITG with an integrated trigram language model. Phrase translation parameters are learned from parallel corpora with approximately 8.5 million words for each of the language pairs. The English language model is trained on the entire corpus of English parliamentary proceedings provided with the Europarl distribution. We report results on the 2000 development test set sentences of length up to 126 words (average length was 30 words).

[2]The software for this clustering technique is available at `http://research.microsoft.com/˜joshuago/`.

[3]We assumed that each cluster had a uniform distribution over all the words in that cluster.

[4]See `http://www.statmt.org/wmt08` for details.

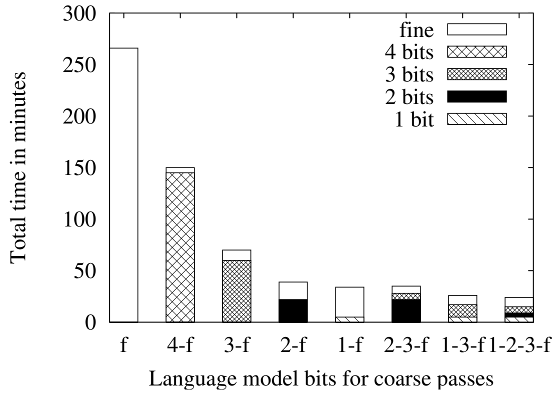


Figure 7: Many passes with extremely simple language models produce the highest speed-ups.

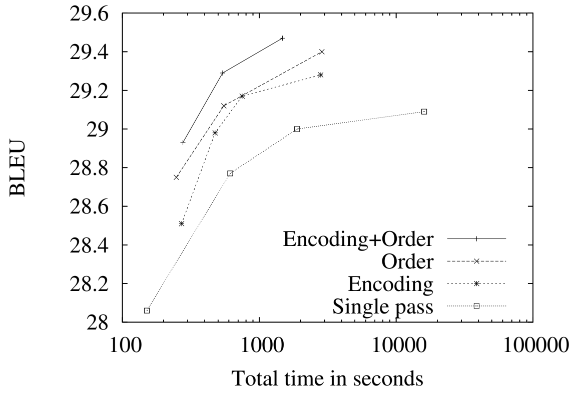


Figure 8: A combination of order-based and encoding-based coarse-to-fine decoding yields the best results.

Our ITG translation model is broadly competitive with state-of-the-art phrase-based-models trained on the same data. For example, on the Europarl development test set, we fall short of Moses (Koehn et al., 2007) by less than one BLEU point. On Spanish-English we get 29.47 BLEU (compared to Moses's 30.40), on French-English 29.34 (vs. 29.95), and 23.80 (vs. 24.64) on German-English. These differences can be attributed primarily to the substantially richer distortion model used by Moses.

The multipass coarse-to-fine architecture that we have introduced presents many choice points. In the following, we investigate various axes individually. We present our findings as BLEU-to-time plots, where the tradeoffs were generated by varying the complexity and the number of coarse passes, as well as the pruning thresholds and beam sizes. Unless otherwise noted, the experiments are on Spanish-English using trigram language models. When different decoder settings are applied to the same model, MERT weights (Och, 2003) from the unprojected single pass setup are used and are kept constant across runs. In particular, the same MERT weights are used for all coarse passes; note that this slightly disadvantages the multipass runs, which use MERT weights optimized for the single pass decoder.

5.1 Clustering

In section Section 4, HMM clustering and JClustering gave lower perplexities than frequency and random clustering when using the same number of bits for encoding the language model. To test how these models perform at pruning, we ran our decoder several times, varying only the clustering source. In each case, we used a 2-bit trigram model as a single coarse pass, followed by a fine output pass. Figure 6 shows that we can obtain significant improvements over the single-pass baseline regardless of the clustering. To no great surprise, HMM clustering and JClustering yield better results, giving a 30-fold speed-up at the same accuracy, or improvements of about 0.3 BLEU when given the same time as the single pass decoder. We discuss this increase in accuracy over the baseline in Section 5.5. Since the performance differences between those two clustering algorithms are negligible, we will use the simpler HMM clustering in all subsequent experiments.

5.2 Spacing

Given a hierarchy of coarse language models, all trigam for the moment, we need to decide on the number of passes and the granularity of the coarse language models used in each pass. Figure 7 shows how decoding time varies for different multipass schemes to achieve the same translation quality. A single coarse pass with a 4-bit language model cuts decoding time almost in half. However, one can further cut decoding time by starting with even coarser language models. In fact, the best results are achieved by decoding in sequence with 1-, 2- and 3-bit language models before running the final fine trigram pass. Interestingly, in this setting, each pass takes about the same amount of time. A similar observation was reported in the parsing literature, where coarse-to-fine inference with multiple passes

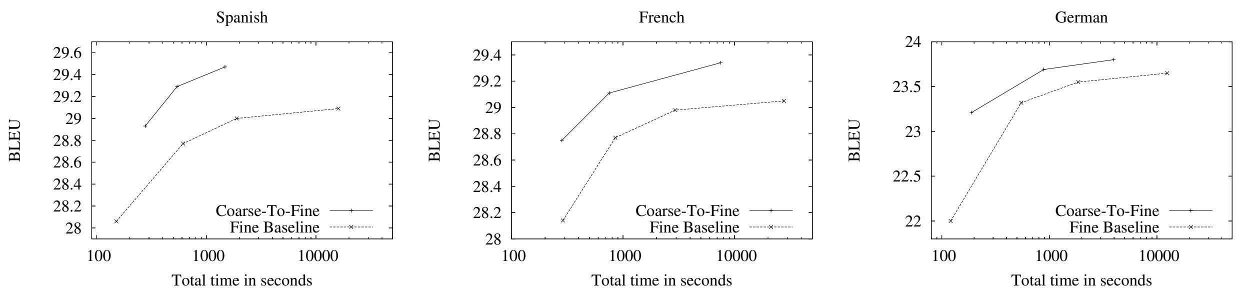


Figure 9: Coarse-to-fine decoding is faster than single pass decoding with a trigram language model and leads to better BLEU scores on all language pairs and for all parameter settings.

of roughly equal complexity produces tremendous speed-ups (Petrov and Klein, 2007).

5.3 Encoding vs. Order

As described in Section 2, the language model complexity can be reduced either by decreasing the vocabulary size (encoding-based projection) or by lowering the language model order from trigram to bigram (order-based projection). Figure 7 shows that both approaches alone yield comparable improvements over the single pass baseline. Fortunately, the two approaches are complimentary, allowing us to obtain further improvements by combining both. We found it best to first do a series of coarse bigram passes, followed by a fine bigram pass, followed by a fine trigram pass.

5.4 Final Results

Figure 9 compares our multipass coarse-to-fine decoder using language refinement to single pass decoding on three different languages. On each language we get significant improvements in terms of efficiency as well as accuracy. Overall, we can achieve up to 50-fold speed-ups at the same accuracy, or alternatively, improvements of 0.4 BLEU points over the best single pass run.

In absolute terms, our decoder translates on average about two Spanish sentences per second at the highest accuracy setting.[5] This compares favorably to the Moses decoder (Koehn et al., 2007), which takes almost three seconds per sentence.

5.5 Search Error Analysis

In multipass coarse-to-fine decoding, we noticed that in addition to computational savings, BLEU scores tend to improve. A first hypothesis is that coarse-to-fine decoding simply improves search quality, where fewer good items fall off the beam compared to a simple fine pass. However, this hypothesis turns out to be incorrect. Table 1 shows the percentage of test sentences for which the BLEU score or log-likelihood changes when we switch from single pass decoding to coarse-to-fine multipass decoding. Only about 30% of the sentences get translated in the same way (if much faster) with coarse-to-fine decoding. For the rest, coarse-to-fine decoding mostly finds translations with lower likelihood, but higher BLEU score, than single pass decoding.[6] An increase of the underlying objectives of interest when pruning despite an increase in model-score search errors has also been observed in monolingual coarse-to-fine syntactic parsing (Charniak et al., 1998; Petrov and Klein, 2007). This effect may be because coarse-to-fine approximates certain minimum Bayes risk objective. It may also be an effect of model intersection between the various passes' models. In any case, both possibilities are often perfectly desirable. It is also worth noting that the number of search errors incurred in the coarse-to-fine approach can be dramatically reduced (at the cost of decoding time) by increasing the pruning thresholds. However, the fortuitous nature of coarse-to-fine search errors seems to be a substantial and desirable effect.

[5]Of course, the time for an average sentence is much lower, since long sentences dominate the overall translation time.

[6]We compared the influence of multipass decoding on the TM score and the LM score; both decrease.

		LL		
		$>$	$=$	$<$
BLEU	$>$	3.6%	-	26.3%
	$=$	1.5%	29.6 %	12.9 %
	$<$	2.2%	-	24.1%

Table 1: Percentage of sentences for which the BLEU score/log-likelihood improves/drops during coarse-to-fine decoding (compared to single pass decoding).

6 Conclusions

We have presented a coarse-to-fine syntactic decoder which utilizes a novel encoding-based language projection in conjunction with order-based projections to achieve substantial speed-ups. Unlike A* methods, a posterior pruning approach allows multiple passes, which we found to be very beneficial for total decoding time. When aggressively pruned, coarse-to-fine decoding can incur additional search errors, but we found those errors to be fortuitous more often than harmful. Our framework applies equally well to other translation systems, though of course interesting new challenges arise when, for example, the underlying SCFGs become more complex.

References

E. Bingham and H.i Mannila. 2001. Random projection in dimensionality reduction: applications to image and text data. In *KDD '01*.

P. Brown, V. Della Pietra, P. deSouza, J. Lai, and R. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*.

E. Charniak, S. Goldwater, and M. Johnson. 1998. Edge-based best-first chart parsing. 6^{th} *Workshop on Very Large Corpora*.

E. Charniak, M. Johnson, D. McClosky, et al. 2006. Multi-level coarse-to-fine PCFG Parsing. In *HLT-NAACL '06*.

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *ACL '05*.

J. Goodman. 1997. Global thresholding and multiple-pass parsing. In *EMNLP '97*.

J. Goodman. 2001. A bit of progress in language modeling. Technical report, Microsoft Research.

A. Haghighi, J. DeNero, and D. Klein. 2007. A* search via approximate factoring. In *NAACL '07*.

L. Huang and D. Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *ACL '07*.

D. Klein and C. Manning. 2003. A* parsing: fast exact viterbi parse selection. In *NAACL '03*.

P. Koehn, H. Hoang, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL '07*.

P. Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*.

I. D. Melamed. 2004. Statistical machine translation by parsing. In *ACL '04*.

F. Och. 2003. Minimum error rate training in statistical machine translation. In *ACL '03*.

S. Petrov and D. Klein. 2007. Improved inference for unlexicalized parsing. In *HLT-NAACL '07*.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *ACL '06*.

A. Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *ICSLP '02*.

A. Venugopal, A. Zollmann, and S. Vogel. 2007. An efficient two-pass approach to synchronous-CFG driven statistical MT. In *HLT-NAACL '07*.

D. Wu. 1996. A polynomial-time algorithm for statistical machine translation. In *ACL '96*.

D. Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. In *Computational Linguistics*.

R. Zens and H. Ney. 2003. A comparative study on reordering constraints in statistical machine translation. In *ACL '03*.

H. Zhang and D. Gildea. 2008. Efficient multi-pass decoding for synchronous context free grammars. In *ACL '08*.

Adding Redundant Features for CRFs-based Sentence Sentiment Classification

Jun Zhao, Kang Liu, Gen Wang

National Laboratory of Pattern Recognition

Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China

{jzhao, kliu, gwang}@nlpr.ia.ac.cn

Abstract

In this paper, we present a novel method based on CRFs in response to the two special characteristics of "contextual dependency" and "label redundancy" in sentence sentiment classification. We try to capture the contextual constraints on sentence sentiment using CRFs. Through introducing redundant labels into the original sentimental label set and organizing all labels into a hierarchy, our method can add redundant features into training for capturing the label redundancy. The experimental results prove that our method outperforms the traditional methods like NB, SVM, MaxEnt and standard chain CRFs. In comparison with the cascaded model, our method can effectively alleviate the error propagation among different layers and obtain better performance in each layer.

1 Introduction[*]

There are a lot of subjective texts in the web, such as product reviews, movie reviews, news, editorials and blogs, etc. Extracting these subjective texts and analyzing their orientations play significant roles in many applications such as electronic commercial, etc. One of the most important tasks in this field is sentiment classification, which can be performed in several levels: word level, sentence level, passage level, etc. This paper focuses on sentence level sentiment classification.

Commonly, sentiment classification contains three layers of sub-tasks. From upper to lower, (1) Subjective/Objective classification: the subjective texts are extracted from the corpus teeming with both subjective and objective texts. (2) Polarity classification: a subjective text is classified into "positive" or "negative" according to the sentimental expressions in the text. (3) Sentimental strength rating: a subjective text is classified into several grades which reflect the polarity degree of "positive" or "negative". It is a special multi-class classification problem, where the classes are ordered. In machine learning, this kind of problem is also regarded as an ordinal regression problem (Wei Wu et al. 2005). In this paper, we mainly focus on this problem in sentiment classification.

Sentiment classification in sentence level has its special characteristics compared with traditional text classification tasks. Firstly, the sentiment of each sentence in a discourse is not independent to each other. In other words, the sentiment of each sentence is related to those of other adjacent sentences in the same discourse. The sentiment of a sentence may vary in different contexts. If we detach a sentence from the context, its sentiment may not be inferred correctly. Secondly, there is redundancy among the sentiment classes,

[*] Contact: Jun ZHAO, jzhao@nlpr.ia.ac.cn

especially in sentimental strength classes. For example:

"I love the scenario of "No country for old man" very much!!"

"This movie sounds good."

The first sentence is labeled as "highly praised" class and the second one is labeled as "something good" class. Both the sentences express positive sentiment for the movie, but the former expresses stronger emotion than the latter. We can see that both "highly praised" and "something good" belong to an implicit class "positive", which can be regarded as the relation between them. If we add these implicit classes in the label set, the sentiment classes will form a hierarchical structure. For example, "positive" can be regarded as the parent class of "highly praised" and "something good", "subjective" can be regarded as the parent class of "positive" and "negative". This implicit hierarchical structure among labels should not be neglected because it may be beneficial for improving the accuracy of sentiment classification. In the paper, we call this characteristic of sentiment classification as "label redundancy". Unfortunately, in our knowledge most of the current research treats sentiment classification as a traditional multi-classification task or an ordinal regression task, which regard the sentimental classes being independent to each other and each sentence is also independent to the adjacent sentences in the context. In other words, they neglect the contextual information and the redundancy among sentiment classes.

In order to consider the contextual information in the process of the sentence sentiment classification, some research defines contextual features and some uses special graph-based formulation, like (Bo Pang, et al. 2005). In order to consider the label redundancy, one potential solution is to use a cascaded framework which can combine subjective/objective classification, polarity classification and sentimental strength classification together, where the classification results of the preceding step will be the input of the subsequent one. However, the subsequent classification cannot provide constraint and correction to the results of the preceding step, which will lead to the accumulation and propagation of the classification errors. As a result, the performance of sentiment analysis of sentences is often not satisfactory.

This paper focuses on the above two special characteristics of the sentiment classification problem in the sentence level. To the first characteristic, we regard the sentiment classification as a sequence labeling problem and use conditional random field (CRFs) model to capture the relation between two adjacent sentences in the context. To the second characteristic, we propose a novel method based on a CRF model, in which the original task is mapped to a classification on a hierarchical structure, which is formed by the original label set and some additional implicit labels. In the hierarchical classification framework, the relations between the labels can be represented as the additional features in classification. Because these features are related to the original labels but unobserved, we name them as "redundant features" in this paper. They can be used to capture the redundant and hierarchical relation between different sentiment classes. In this way, not only the performance of sentimental strength rating is improved, the accuracies of subjective/objective classification and polarity classification are also improved compared with the traditional sentiment classification method. And in comparison with the cascaded method, the proposed approach can effectively alleviate error propagation. The experimental results on movie reviews prove the validity of our method.

2 Capturing Contextual Influence for Sentiment Classification

For capturing the influence of the contexts to the sentiment of a sentence, we treat original sentiment classification as a sequence labeling problem. We regard the sentiments of all the sentences throughout a paragraph as a sequential flow of sentiments, and we model it using a conditional model. In this paper, we choose Conditional Random Fields (CRFs) (Lafferty et al, 2001) because it has better performance than other sequence labeling tools in most NLP applications.

CRFs are undirected graphical models used to calculate the conditional probability of a set of labels given a set of input variables. We cite the definitions of CRFs in (Lafferty et al, 2001). It defines the conditional probability proportional to the product of potential functions on cliques of the graph,

$$P_\lambda(Y \mid X) = \frac{\exp \lambda \cdot F(Y,X)}{Z(X)} \quad (1)$$

where X is a set of input random variables and Y is a set of random labels. $F(Y,X)$ is an arbitrary feature function over its arguments, λ is a learned weight for each feature function and $Z(X) = \sum_y \exp(\lambda \cdot F(Y,X))$.

The training of CRFs is based on Maximum Likelihood Principle (Fei Sha et al. 2003). The log likelihood function is

$$L(\lambda) = \sum_k \left[\lambda \cdot F(Y_k, X_k) - \log Z_\lambda(X_k)\right]$$

Therefore, Limited-memory BFGS (L-BFGS) algorithm is used to find this nonlinear optimization parameters.

3 Label Redundancy in Sentiment Classification

In this section, we explain the "label redundancy" in sentiment classification mentioned in the first section. We will analyze the effect of the label redundancy on the performance of sentiment classification from the experimental view.

We conduct the experiments of polarity classification and sentimental strength rating on the corpus which will be introduced in section 5 later. The class set is also illustrated in that section. Polarity classification is a three-class classification process, and sentimental strength rating is a five-class classification process. We use first 200 reviews as the training set which contains 6,079 sentences, and other 49 reviews, totally 1,531 sentences, are used as the testing set. Both the three-class classification and the five-class classification use standard CRFs model with the same feature set. The results are shown in Table 1, 2 and 3, where "Answer" denotes the results given by human, "Results" denotes the results given by CRFs model，"Correct" denotes the number of correct samples which is labeled by CRFs model. We use precision, recall and F1 value as the evaluation metrics.

Table 1 gives the result of sentimental strength rating. Table 2 shows the polarity classification results extracted from the results of sentimental strength rating in Table 1. The extraction process is as follows. In the sentimental strength rating results, we combine the sentences with "PP" class and the sentences with "P" class into "Pos" class, and the sentences with "NN" class and the sentences with "N" class into "Neg" class. So the results of five-class classification are transformed into the results of three-class classification. Table 3 is the results of performing polarity classification in the data set by CRFs directly.

Label	Answer	Results	Correct	Precision	Recall	F1
PP	51	67	5	0.0746	0.0980	0.0847
P	166	177	32	0.1808	0.1928	0.1866
Neu	1190	1118	968	0.8658	0.81.34	0.8388
N	105	140	25	0.1786	0.2381	0.2041
NN	19	29	1	0.0345	0.0526	0.0417
Total	1531	1531	1031	0.67.34	0.6734	0.6734

Table 1. Result of Sentimental Strength Rating

Label	Answer	Results	Correct	Precision	Recall	F1
Pos	217	244	79	0.3238	0.3641	0.3427
Neu	1190	1118	968	0.8658	0.8134	0.8388
Neg	124	169	41	0.2426	0.3306	0.2799
Total	1531	1531	1088	0.7106	0.7106	0.7106

Table 2. Result of Polarity Classification Extracted from Table 1.

Label	Answer	Results	Correct	Precision	Recall	F1
Pos	217	300	108	0.3600	0.4977	0.4178
Neu	1190	1101	971	0.8819	0.8160	0.8477
Neg	124	130	40	0.3077	0.3226	0.3150
Total	1531	1531	1119	0.7309	0.7309	0.7309

Table 3. Result of Polarity Classification

From the results we can find the following phenomena.

(1) The corpus is severely unbalanced, the objective sentences take the absolute majority in the corpus, which leads to the poor accuracy for classifying subjective sentences. The experiment in Table 1 puts polarity classification and sentimental strength rating under a unique CRFs model, without considering the redundancy and hierarchical structure between different classes. As a result, the features for polarity classification will usually cover the features for sentimental strength rating. These reasons can explain why there is only one sample labeled as “NN” correctly and only 5 samples labeled as “PP” correctly.

(2) Comparing Table 2 with 3, we can find that, the F1 value of the polarity classification results extracted from sentimental strength rating results is lower than that of directly conducting polarity classification. That is because the redundancy between sentimental strength labels makes the classifier confused to determine the polarity of the sentence. Therefore, we should deal with the sentiment analysis in a hierarchical frame which can consider the redundancy between the different classes and make full use of the subjective and polarity information implicitly contained in sentimental strength classes.

4 Capturing Label Redundancy for CRFs via Adding Redundant Features

As mentioned above, it’s important for a classifier to consider the redundancy between different labels. However, from the standard CRFs described in formula (1), we can see that the training of CRFs only maximizes the probabilities of the observed labels Y in the training corpus. Actually, the redundant relation between sentiment labels is unobserved. The standard CRFs still treats each class as an isolated item so that its performance is not satisfied.

In this section, we propose a novel method for sentiment classification, which can capture the redundant relation between sentiment labels through adding redundant features. In the following, we firstly show how to add these redundant features, then illustrate the characteristics of this method. After that, for the sentiment analysis task, the process of feature generation will be presented.

4.1 Adding Redundant Features for CRFs

Adding redundant features has two steps. Firstly, an implicit redundant label set is designed, which can form a multi-layer hierarchical structure together with the original labels. Secondly, in the hierarchical classification framework, the implicit labels, which reflect the relations between the original labels, can be used as redundant features in the training process. We will use the following example to illustrate the first step for sentimental strength rating task.

For the task of sentimental strength rating, the original label set is {“PP (highly praised)”, “P (something good)”, “Neu (objective description)”, “N (something that needs improvement)” and “NN (strong aversion)”}. In order to introduce redundant labels, the 5-class classification task is decomposed into the following three layers shown in Figure 1. The label set in the first layer is {“subjective”, “objective”}, The label set in the second layer is for polarity classification {“positive”, “objective”, “negative”}, and the label set in the third layer is the original set. Actually, the labels in the first and second layers are unobserved redundant labels, which will not be reflected in the final classification result obviously.

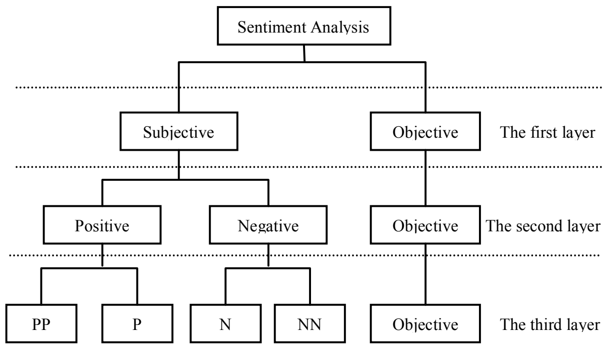


Figure 1. The hierarchical structure of sentimental labels

In the second step, with these redundant labels, some implicit features can be generated for CRFs. So the standard CRFs can be rewritten as follows.

$$P(T \mid X) = \frac{\exp(F(X,T)\cdot\lambda)}{Z_T(X)} = \frac{\exp(\sum_{j=1}^{m} F_j(X,Y_j)\cdot\lambda_j)}{\sum_T \exp(\sum_{j=1}^{m} F_j(X,Y_j)\cdot\lambda_j)} \quad (2)$$

where $T=(Y_1,Y_2,...Y_j...,Y_m)$, and Y_j denotes the label sequence in the j^{th} layer. $F_j(X,Y_j)$ denotes the arbitrary feature function in the j^{th} layer.

From the formula (2), we can see that the original label set is rewritten as $T=(Y_1,Y_2,...Y_j...,Y_m)$, which contains implicit labels in the hierarchical structure shown in Figure 1. The difference between our method and the standard chain CRFs is that we make some implicit redundant features to be active when training. The original feature function $F(Y,X)$ is replaced by $\sum_{j=1}^{m} F_j(X,Y_j)$. We use an example to illustrate the process of feature generation. When a sentence including the word "good" is labeled as "PP", our model not only generate the state feature (good, "PP"), but also two implicit redundant state feature (good, "positive") and (good, "subjective"). Through adding larger-granularity labels "positive" and "negative" into the model, our method can increase the probability of "positive" and decrease the probability of "negative". Furthermore, "P" and "PP" will share the probability gain of "positive", therefore the probability of "P" will be larger than that of "N". For the transition feature, the same strategy is used. Therefore the complexity of its training procedure is $O(M \times N \times \sum_j^m F_j \times l)$ where M is the number of the training samples, N is the average sentence length, F_j is the average number of activated features in the j^{th} layer, l is the number of the original labels and m is the number of the layers. For the complexity of the decoding procedure, our method has $O(N \times \sum_j^m F_j \times l)$.

It's worth noting that, (1) transition features are extracted in each layer separately rather than across different layers. For example, feature (good, "subjective", "positive") will never be extracted because "subjective" and "positive" are from different layers; (2) if one sentence is labeled as "Neu", no implicit redundant features will be generated.

4.2 The Characteristics of Our Method

Our method allows that the label sets are dependent and redundant. As a result, it can improve the performance of not only the classifier for the original sentimental strength rating task, but also the classifiers for other tasks in the hierarchical frame, i.e. polarity classification and subjective/objective classification. This kind of dependency and redundancy can lead to two characteristics of the proposed method for sentiment classification compared with traditional methods, such as the cascaded method.

(1) Error-correction: Two dependent tasks in the neighboring layers can correct the errors of each other relying on the inconsistent redundant information. For example, if in the first layer, the features activated by "objective" get larger scores than the features activated by "subjective", and in the second layer the features activated by "positive" get larger scores than the features activated by "objective", then inconsistency emerges. At this time, our method can globally select the label with maximum probability. This characteristic can make up the deficiency of the cascaded method which may induce error propagation.

(2) Differentiating the ordinal relation among sentiment labels: Our method organizes the ordinal sentiment labels into a hierarchy through introducing redundant labels into standard chain CRFs, in this way the degree of classification errors can be controlled. In the different layers of sentiment analysis task, the granularities of classification are different. Therefore, when an observation cannot be correctly labeled on a smaller-granularity label set, our method will use the larger-granularity labels in the upper layer to control the final classification labels.

4.3 Feature Selection in Different Layers

For feature selection, our method selects different features for each layer in the hierarchical frame.

In the top layer of the frame shown in Figure 1, for subjective/objective classification task, we use not only adjectives and the verbs which contain subjective information (e.g., "believe", "think") as the features, but also the topic words. The topic words are defined as the nouns or noun phases which frequently appear in the corpus. We believe that some topic words contain subjective information.

In the middle and bottom layers, we not only use the features in the first layer, but also some special features as follows.

(1) The prior orientation scores of the sentiment words: Firstly, a sentiment lexicon is generated by extending the synonymies and antonyms in WordNet[2] from a positive and negative seed list. Then, the positive score and the negative score of a sentiment word are individually accumulated and weighted according to the polarity of its synonymies and antonyms. At last we scale the normalized distance of the two scores into 5 levels, which will be the prior orientation of the word. When there is a negative word, like {not, no, can't, merely, never, …}, occurring nearby the feature word in the range of 3 words size window, the orientation of this word will be reversed and "NO" will be added in front of the original feature word for creating a new feature word.

(2) Sentence transition features: We consider two types of sentence transition features. The first type is the conjunctions and the adverbs occurring in the beginning of this sentence. These conjunctions and adverbs are included in a word list which is manually selected, like {and, or, but, though, however, generally, contrarily, …}. The second type of the sentence transition feature is the position of the sentence in one review. The reason lies in that: the reviewers often follow some writing patterns, for example some reviewers prefer to concede an opposite factor before expressing his/her real sentiment. Therefore, we divide a review into five parts, and assign each sentence with the serial number of the part which the sentence belongs to.

5 Experiments

5.1 Data and Baselines

In order to evaluate the performance of our method, we conducted experiments on a sentence level annotation corpus obtained from Purdue University, which is also used in (Mao and Lebanon 07). This corpus contains 249 movie reviews and 7,610 sentences totally, which is randomly selected from the Cornell sentence polarity dataset v1.0. Each sentence was hand-labeled with one of five classes: PP (highly praised), P (something good), Neu (objective description), N (something that needs improvement) and NN (strong aversion), which contained the orientation polarity of each sentence. Based on the 5-class manually labeled results mentioned above, we also assigned each sentence with one of three classes: Pos (positive polarity), Neu (objective description), Neg (negative polarity). Data statistics for the corpus are given in Table 4.

Label	Pos		Neu	Neg		Total
	PP	P	Neu	N	NN	
5 classes	383	860	5508	694	165	7610
3 classes	1243		5508	859		7610

Table 4. Data Statistics for Movies Reviews Corpus

There is a problem in the dataset that more than 70% of the sentences are labeled as "Neu" and labels are seriously unbalanced. As a result, the "Neu" label is over-emphasized. For this problem, Mao and Lebanon (2007) made a balanced data set (equal number sentences for different labels) which is sampled in the original corpus. Since randomly sampling sentences from the original corpus will break the intrinsic relationship between two adjacent sentences in the context, we don't create balanced label data set.

For the evaluation of our method, we choose accuracy as the evaluation metrics and some classical methods as the baselines. They are Naïve Bayes (NB), Support Vector Machine (SVM), Maximum Entropy (MaxEnt) (Kamal Nigam et al. 1999) and standard chain CRFs (Fei et al. 2003). We also regard cascaded-CRFs as our baseline for comparing our method with the cascaded-based method. For NB, we use Laplace smoothing method. For SVM, we use the LibSVM[3] with a linear kernel function[4]. For MaxEnt, we use the implementation in the toolkit *Mallet*[5]. For CRFs,

[2] http://wordnet.princeton.edu/

[3] http://www.csie.ntu.tw/~cjlin/libsvm

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

[5] http://mallet.cs.umass.edu/index.php/Main_Page

Label	NB	SVM	MaxEnt	Standard CRF	Cascaded CRF	Our Method
PP	0.1745	0.2219	0.2055	0.2027	**0.2575**	0.2167
P	0.2049	0.2877	0.2353	0.2536	0.2881	**0.3784**
Neu	0.8083	**0.8685**	0.8161	0.8273	0.8554	0.8269
N	0.2636	0.3014	0.2558	0.2981	0.3092	**0.4204**
NN	0.0976	0.1162	0.1148	0.1379	0.1510	**0.2967**
Total	0.6442	0.6786	0.6652	0.6856	0.7153	**0.7521**

Table 5. The accuracy of Sentimental Strength Rating

Label	NB	SVM	MaxEnt	Standard CRF	Cascaded-CRF	Our Method
Pos	0.4218	0.4743	0.4599	0.4405	0.5122	**0.6008**
Neu	0.8147	0.8375	0.8424	0.8260	**0.8545**	0.8269
Neg	0.3217	0.3632	0.2739	0.3991	0.4067	**0.5481**
Total	0.7054	0.7322	0.7318	0.7327	0.7694	**0.7855**

Table 6．The Results of Polarity Classification

Label	NB	SVM	MaxEnt	Standard CRF	Our Method
Subjective	0.4743	0.5847	0.4872	0.5594	**0.6764**
Objective	0.8170	0.8248	0.8212	**0.8312**	0.8269
Total	0.7238	0.7536	0.7518	0.7561	**0.8018**

Table 7. The accuracy of Subjective/Objective Classification

we use the implementation in Flex-CRFs[6]. We set the iteration number to 120 in the training process of the method based on CRFs. In the cascaded model we set 3 layers for sentimental strength rating, where the first layer is subjective/objective classification, the second layer is polarity classification and the last layer is sentimental strength classification. The upper layer passes the results as the input to the next layer.

5.2 Sentimental Strength Rating

In the first experiment, we evaluate the performance of our method for sentimental strength rating. Experimental results for each method are given in Table 5. We not only give the overall accuracy of each method, but also the performance for each sentimental strength label. All baselines use the same feature space mentioned in section 4.3, which combine all the features in the three layers together, except cascaded CRFs and our method. In cascaded-CRFs and our method, we use different features in different layers mentioned in section 4.3. These results were gathered using 5-fold cross validation with one fold for testing and the other 4 folds for training.

From the results, we can obtain the following conclusions. (1) The three versions of CRFs perform consistently better than Naïve Bayes, SVM and MaxEnt methods. We think that is because CRFs model considers the contextual influence of each sentence. (2) Comparing the performance of cascaded CRFs with that of standard sequence CRFs, we can see that not only the overall accuracy but also the accuracy for each sentimental strength label are improved, where the overall accuracy is increased by 3%. It proves that taking the hierarchical relationship between labels into account is very essential for sentiment classification. The reason is that: the cascaded model performs sentimental strength rating in three hierarchical layers, while standard chain CRFs model treats each label as an independent individual. So the performance of the cascaded model is superior to the standard chain CRFs. (3) The experimental results also show that our method performs better than the Cascaded CRFs. The classification accuracy is improved from 71.53% to 75.21%. We think that is because our method adds the label redundancy among the sentimental strength labels into consideration through adding redundant features into the feature sets, and the three subtasks in the cascaded model are merged into a unified model. So the output result is a global optimal result. In this way, the problem of error propagation in the cascaded frame can be alleviated.

[6] http://flexcrfs.sourceforge.net

5.3 Sentiment Polarity Classification

In the second experiment, we evaluate the performance of our method for sentiment polarity classification. Our method is based on a hierarchical frame, which can perform different tasks in different layers at the same time. For example, it can determine the polarity of sentences when sentimental strength rating is performed. Here, the polarity classification results of our method are extracted from the results of the sentimental strength rating mentioned above. In the sentimental strength rating results, we combine the sentences with PP label and the sentences with P label into one set, and the sentences with NN label and the sentences with N label into one set. So the results of 5-class classification are transformed into the results of 3-class classification. Other methods like NB, SVM, MaxEnt, standard chain CRFs perform 3-class classification directly, and their label sets in the training corpus is {Pos, Neu, Neg}. The parameter setting is the same as sentimental strength rating. For the cascaded-CRFs method, we firstly perform subjective/objective classification, and then determine the polarity of the sentences based on the subjective sentences. The experimental results are given in Table 6.

From the experimental results, we can obtain the following conclusion for sentiment polarity classification, which is similar to the conclusion for sentimental strength rating mentioned in section 5.2. That is both our model and the cascaded model can get better performance than other traditional methods, such as NB, SVM, MaxEnt, etc. But the performance of the cascaded CRFs (76.94%) is lower than that of our method (78.55%). This indicates that because our method exploits the label redundancy in the different layers, it can increase the accuracies of both polarity classification and sentimental strength rating at the same time compared with other methods.

5.4 Subjective/Objective Classification

In the last experiment, we test our method for subjective/objective classification. The subjective/objective label of the data is extracted from its original label like section 5.3. As the same as the experiment for polarity classification, all baselines perform subjective/objective classification directly. It's no need to perform the cascaded-based method because it's a 2-class task. The results of our method are extracted from the results of the sentimental strength rating too. The results are shown in Table 7. From it, we can obtain the similar conclusion, i.e. our method outperforms other methods and has the 80.18% classification accuracy. Our method, which introduces redundant features into training, can increase the accuracies of all tasks in the different layers at the same time compared with other baselines. It proves that considering label redundancy are effective for promoting the performance of a sentimental classifier.

6 Related Works

Recently, many researchers have devoted into the problem of the sentiment classification. Most of researchers focus on how to extract useful textual features (lexical, syntactic, punctuation, etc.) for determining the semantic orientation of the sentences using machine learning algorithm (Bo et al. 2002; Kim and Hovy, 2004; Bo et al. 2005, Hu et al. 2004; Alina et al 2008; Alistair et al 2006). But fewer researchers deal with this problem using CRFs model.

For identifying the subjective sentences, there are several research, like (Wiebe et al, 2005). For polarity classification on sentence level, (Kim and Hovy, 2004) judged the sentiment by classifying a pseudo document composed of synonyms of indicators in one sentence. (Pang and Lee, 04) proposed a semi-supervised machine learning method based on subjectivity detection and minimum-cut in graph.

Cascaded models for sentiment classification were studied by (Pang and Lee, 2005). Their work mainly used the cascaded frame for determining the orientation of a document and the sentences. In that work, an initial model is used to determine the orientation of each sentence firstly, then the top subjective sentences are input into a document - level model to determine the document's orientation.

The CRFs has previously been used for sentiment classification. Those methods based on CRFs are related to our work. (Mao et al, 2007) used a sequential CRFs regression model to measure the polarity of a sentence in order to determine the sentiment flow of the authors in reviews. However, this method must manually select a word set for constraints, where each selected word achieved the highest correlation with the sentiment. The performance of isotonic CRFs is strongly related to the selected word set. (McDonald et al 2007; Ivan et al 2008) proposed a structured model based on CRFs for jointly classifying the sentiment of text at varying levels of granularity. They put the sentence level and document level sentiment analysis in an integrated model and employ the orientation of the document to influence the decision of sentence's orientation. Both the above two methods didn't consider the redundant and hierarchical relation between sentimental strength labels. So their methods cannot get better results for the problem mentioned in this paper.

Another solution to this problem is to use a joint multi-layer model, such as dynamic CRFs, multi-layer CRFs, etc. Such kind of models can treat the three sub-tasks in sentiment classification as a multi-task problem and can use a multi-layer or hierarchical undirected graphic to model the sentiment of sentences. The main difference between our method and theirs is that we consider the problem from the feature representation view. Our method expands the feature set according to the number of layers in the hierarchical frame. So the complexity of its decoding procedure is lower than theirs, for example the complexity of the multi-layer CRFs is $O(N \times F \times \prod_j l_j)$ when decoding and our method only has $O(N \times \sum_j F_j \times l)$, where N is the average sentence length, F_j is the average number of activated features in the j^{th} layer, l is the number of the original labels.

7 Conclusion and Future Work

In the paper, we propose a novel method for sentiment classification based on CRFs in response to the two special characteristics of "contextual dependency" and "label redundancy" in sentence sentiment classification. We try to capture the contextual constraints on the sentence sentiment using CRFs. For capturing the label redundancy among sentiment classes, we generate a hierarchical framework through introducing redundant labels, under which redundant features can be introduced. The experimental results prove that our method outperforms the traditional methods (like NB, SVM, ME and standard chain CRFs). In comparison with cascaded CRFs, our method can effectively alleviate error propagation among different layers and obtain better performance in each layer.

For our future work, we will explore other hierarchical models for sentimental strength rating because the experiments presented in this paper prove this hierarchical frame is effective for ordinal regression. We would expand the idea in this paper into other models, such as Semi-CRFs and Hierarchical-CRFs.

Acknowledgments

The work is supported by the National Natural Science Foundation of China under Grants no. 60673042, the Natural Science Foundation of Beijing under Grants no. 4073043 and the National High Technology Development 863 Program of China under Grants no. 2006AA01Z144.

References

Alina A. and Sabine B. 2008. *When Specialists and Generalists Work Together: Overcoming Domain Dependence in Sentiment Tagging*. In *Proc. of ACL-08*

Alistair Kennedy and Diana Inkpen. 2006. *Sentiment Classification of Movie Reviews Using Contextual Valence Shifter*s. *Computational Intelligence*, 22(2), pages 110-125

Bo Pang, Lillian Lee and Shivakumar Vaithyanathan. 2002. *Thumbs up? Sentiment classification using machine learning techniques*. In *Proceedings of EMNLP 2002,* pp.79-86.

Bo Pang and Lillian Lee. 2004. *A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts.* In *Proceedings of ACL 2004*, pp.271-278.

Bo Pang and Lillian Lee. 2005. *Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales*. In *Proceedings of ACL 2005*, pp.115-124.

Ivan Titov and Ryan McDonald. 2008. *A Joint Model o f Text and Aspect Ratings of Sentiment Summarization.* In *Proceedings of ACL-08*, pages 308-316

Janyce Webie, Theresa Wilson and Claire Cardie. 2005. *Annotating expressions of opinions and emotions in lauguage. Language Resources and Evaluation 2005*

Fei Sha and Fernando Pereira, 2003 *Shallow Parsing with Conditional Random Fields*, In *Proceedings ofHLT-NAACL 2003, Edmonton, Canada*, pp. 213-220.

Kim, S and Edward H. Hovy. 2004. *Determining the Sentiment of Opinions*. In *Proceedings of COLING-04*.

Kamal Nigam, John Lafferty and Andrew McCallum. 1999. *Using Maximum Entropy for Text Classification*. In *Proceedings of IJCAI Workshop on Machine Learning for Information Filtering*, pages 61-67.

J Lafferty, A McCallum, F Pereira. 2001. *Conditional random fields: Probabilistic models for segmenting and labeling sequence data*. In *Proceedings of ICML-01*, pages 282.289.

L. Zhuang, F. Jing, and X.Y. Zhu. 2006. *Movie review mining and summarization*. In *Proceedings of the 15th ACM international conference on Information and knowledge management (CIKM)*, pages 43-50.

M. Hu and B. Liu. 2004a. *Mining and summarizing customer reviews*. In *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168-177.

Ryan McDonald, Kerry Hannan and Tyler Neylon et al. *Structured Models for Fine-to-Coarse Sentiment Analysis*. In *Proceedings of ACL 2007*, pp. 432-439.

Wei Wu, Zoubin Ghahraman, 2005. *Gaussian Processes for Oridinal Regression. The Journal of Machine learning Research*, 2005

Y. Mao and G. Lebanon, 2007. *Isotonic Conditional Random Fields and Local Sentiment Flow. Advances in Neural Information Processing Systems* 19, 2007

Multilingual Subjectivity Analysis Using Machine Translation

Carmen Banea and **Rada Mihalcea**
University of North Texas
carmenb@unt.edu, rada@cs.unt.edu

Janyce Wiebe
University of Pittsburgh
wiebe@cs.pitt.edu

Samer Hassan
University of North Texas
samer@unt.edu

Abstract

Although research in other languages is increasing, much of the work in subjectivity analysis has been applied to English data, mainly due to the large body of electronic resources and tools that are available for this language. In this paper, we propose and evaluate methods that can be employed to transfer a repository of subjectivity resources across languages. Specifically, we attempt to leverage on the resources available for English and, by employing machine translation, generate resources for subjectivity analysis in other languages. Through comparative evaluations on two different languages (Romanian and Spanish), we show that automatic translation is a viable alternative for the construction of resources and tools for subjectivity analysis in a new target language.

1 Introduction

We have seen a surge in interest towards the application of automatic tools and techniques for the extraction of opinions, emotions, and sentiments in text (*subjectivity*). A large number of text processing applications have already employed techniques for automatic subjectivity analysis, including automatic expressive text-to-speech synthesis (Alm et al., 2005), text semantic analysis (Wiebe and Mihalcea, 2006; Esuli and Sebastiani, 2006), tracking sentiment timelines in on-line forums and news (Lloyd et al., 2005; Balog et al., 2006), mining opinions from product reviews (Hu and Liu, 2004), and question answering (Yu and Hatzivassiloglou, 2003).

A significant fraction of the research work to date in subjectivity analysis has been applied to English, which led to several resources and tools available for this language. In this paper, we explore multiple paths that employ machine translation while leveraging on the resources and tools available for English, to automatically generate resources for subjectivity analysis for a new target language. Through experiments carried out with automatic translation and cross-lingual projections of subjectivity annotations, we try to answer the following questions.

First, assuming an English corpus manually annotated for subjectivity, can we use machine translation to generate a subjectivity-annotated corpus in the target language? Second, assuming the availability of a tool for automatic subjectivity analysis in English, can we generate a corpus annotated for subjectivity in the target language by using automatic subjectivity annotations of English text and machine translation? Finally, third, can these automatically generated resources be used to effectively train tools for subjectivity analysis in the target language?

Since our methods are particularly useful for languages with only a few electronic tools and resources, we chose to conduct our initial experiments on Romanian, a language with limited text processing resources developed to date. Furthermore, to validate our results, we carried a second set of experiments on Spanish. Note however that our methods do not make use of any target language specific knowledge, and thus they are applicable to any other language as long as a machine translation engine exists between the selected language and English.

2 Related Work

Research in sentiment and subjectivity analysis has received increasingly growing interest from the natural language processing community, particularly motivated by the widespread need for opinion-based applications, including product and movie reviews, entity tracking and analysis, opinion summarization, and others.

Much of the work in subjectivity analysis has been applied to English data, though work on other languages is growing: e.g., Japanese data are used in (Kobayashi et al., 2004; Suzuki et al., 2006; Takamura et al., 2006; Kanayama and Nasukawa, 2006), Chinese data are used in (Hu et al., 2005), and German data are used in (Kim and Hovy, 2006). In addition, several participants in the Chinese and Japanese Opinion Extraction tasks of NTCIR-6 (Kando and Evans, 2007) performed subjectivity and sentiment analysis in languages other than English.

In general, efforts on building subjectivity analysis tools for other languages have been hampered by the high cost involved in creating corpora and lexical resources for a new language. To address this gap, we focus on leveraging resources already developed for one language to derive subjectivity analysis tools for a new language. This motivates the direction of our research, in which we use machine translation coupled with cross-lingual annotation projections to generate the resources and tools required to perform subjectivity classification in the target language.

The work closest to ours is the one reported in (Mihalcea et al., 2007), where a bilingual lexicon and a manually translated parallel text are used to generate the resources required to build a subjectivity classifier in a new language. In that work, we found that the projection of annotations across parallel texts can be successfully used to build a corpus annotated for subjectivity in the target language. However, parallel texts are not always available for a given language pair. Therefore, in this paper we explore a different approach where, instead of relying on manually translated parallel corpora, we use machine translation to produce a corpus in the new language.

3 Machine Translation for Subjectivity Analysis

We explore the possibility of using machine translation to generate the resources required to build subjectivity annotation tools in a given target language. We focus on two main scenarios. First, assuming a corpus manually annotated for subjectivity exists in the source language, we can use machine translation to create a corpus annotated for subjectivity in the target language. Second, assuming a tool for automatic subjectivity analysis exists in the source language, we can use this tool together with machine translation to create a corpus annotated for subjectivity in the target language.

In order to perform a comprehensive investigation, we propose three experiments as described below. The first scenario, based on a corpus manually annotated for subjectivity, is exemplified by the first experiment. The second scenario, based on a corpus automatically annotated with a tool for subjectivity analysis, is subsequently divided into two experiments depending on the direction of the translation and on the dataset that is translated.

In all three experiments, we use English as a source language, given that it has both a corpus manually annotated for subjectivity (MPQA (Wiebe et al., 2005)) and a tool for subjectivity analysis (OpinionFinder (Wiebe and Riloff, 2005)).

3.1 Experiment One: Machine Translation of Manually Annotated Corpora

In this experiment, we use a corpus in the source language manually annotated for subjectivity. The corpus is automatically translated into the target language, followed by a projection of the subjectivity labels from the source to the target language. The experiment is illustrated in Figure 1.

We use the MPQA corpus (Wiebe et al., 2005), which is a collection of 535 English-language news articles from a variety of news sources manually annotated for subjectivity. Although the corpus was originally annotated at clause and phrase level, we use the sentence-level annotations associated with the dataset (Wiebe and Riloff, 2005). From the total of 9,700 sentences in this corpus, 55% of the sentences are labeled as subjective while the rest are objective. After the automatic translation of the cor-

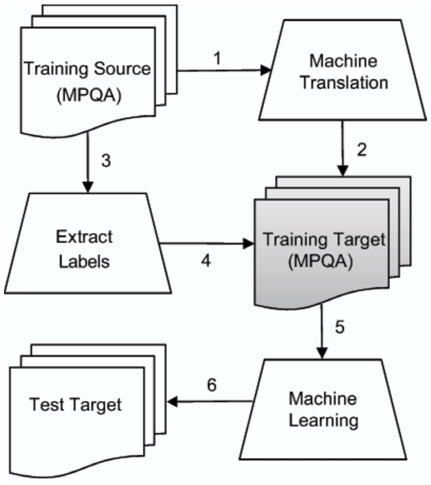


Figure 1: Experiment one: machine translation of manually annotated training data from source language into target language

pus and the projection of the annotations, we obtain a large corpus of 9,700 subjectivity-annotated sentences in the target language, which can be used to train a subjectivity classifier.

3.2 Experiment Two: Machine Translation of Source Language Training Data

In the second experiment, we assume that the only resources available are a tool for subjectivity annotation in the source language and a collection of raw texts, also in the source language. The source language text is automatically annotated for subjectivity and then translated into the target language. In this way, we produce a subjectivity annotated corpus that we can use to train a subjectivity annotation tool for the target language. Figure 2 illustrates this experiment.

In order to generate automatic subjectivity annotations, we use the OpinionFinder tool developed by (Wiebe and Riloff, 2005). OpinionFinder includes two classifiers. The first one is a rule-based high-precision classifier that labels sentences based on the presence of subjective clues obtained from a large lexicon. The second one is a high-coverage classifier that starts with an initial corpus annotated using the high-precision classifier, followed by several bootstrapping steps that increase the size of the lexicon and the coverage of the classifier. For most of our experiments we use the high-coverage classifier.

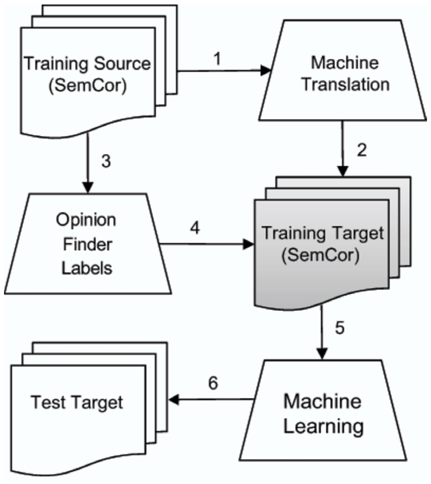


Figure 2: Experiment two: machine translation of raw training data from source language into target language

Table 1 shows the performance of the two OpinionFinder classifiers as measured on the MPQA corpus (Wiebe and Riloff, 2005).

	P	R	F
high-precision	86.7	32.6	47.4
high-coverage	79.4	70.6	74.7

Table 1: Precision (P), Recall (R) and F-measure (F) for the two OpinionFinder classifiers, as measured on the MPQA corpus

As a raw corpus, we use a subset of the SemCor corpus (Miller et al., 1993), consisting of 107 documents with roughly 11,000 sentences. This is a balanced corpus covering a number of topics in sports, politics, fashion, education, and others. The reason for working with this collection is the fact that we also have a manual translation of the SemCor documents from English into one of the target languages used in the experiments (Romanian), which enables comparative evaluations of different scenarios (see Section 4).

Note that in this experiment the annotation of subjectivity is carried out on the original source language text, and thus expected to be more accurate than if it were applied on automatically translated text. However, the training data in the target language is produced by automatic translation, and thus likely to contain errors.

3.3 Experiment Three: Machine Translation of Target Language Training Data

The third experiment is similar to the second one, except that we reverse the direction of the translation. We translate raw text that is available in the target language into the source language, and then use a subjectivity annotation tool to label the automatically translated source language text. After the annotation, the labels are projected back into the target language, and the resulting annotated corpus is used to train a subjectivity classifier. Figure 3 illustrates this experiment.

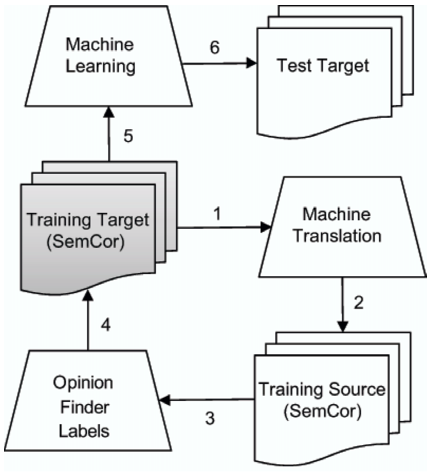


Figure 3: Experiment three: machine translation of raw training data from target language into source language

As before, we use the high-coverage classifier available in OpinionFinder, and the SemCor corpus. We use a manual translation of this corpus available in the target language.

In this experiment, the subjectivity annotations are carried out on automatically generated source text, and thus expected to be less accurate. However, since the training data was originally written in the target language, it is free of translation errors, and thus training carried out on this data should be more robust.

3.4 Upper bound: Machine Translation of Target Language Test Data

For comparison purposes, we also propose an experiment which plays the role of an upper bound on the methods described so far. This experiment involves the automatic translation of the test data from the target language into the source language. The source language text is then annotated for subjectivity using OpinionFinder, followed by a projection of the resulting labels back into the target language.

Unlike the previous three experiments, in this experiment we only generate subjectivity-annotated *resources*, and we do not build and evaluate a standalone subjectivity analysis *tool* for the target language. Further training of a machine learning algorithm, as in experiments two and three, is required in order to build a subjectivity analysis tool. Thus, this fourth experiment is an evaluation of the *resources* generated in the target language, which represents an upper bound on the performance of any machine learning algorithm that would be trained on these resources. Figure 4 illustrates this experiment.

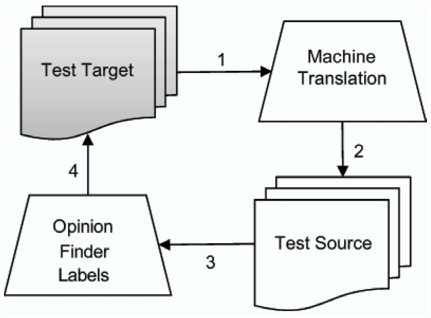


Figure 4: Upper bound: machine translation of test data from target language into source language

4 Evaluation and Results

Our initial evaluations are carried out on Romanian. The performance of each of the three methods is evaluated using a dataset manually annotated for subjectivity. To evaluate our methods, we generate a Romanian training corpus annotated for subjectivity on which we train a subjectivity classifier, which is then used to label the test data.

We evaluate the results against a gold-standard corpus consisting of 504 Romanian sentences manually annotated for subjectivity. These sentences represent the manual translation into Romanian of a small subset of the SemCor corpus, which was removed from the training corpora used in experiments two and three. This is the same evaluation dataset as used in (Mihalcea et al., 2007). Two Romanian native speakers annotated the sentences individually, and the differences were adjudicated through discussions. The agreement of the two annotators is 0.83% (κ = 0.67); when the uncertain annotations are removed, the agreement rises to 0.89 (κ = 0.77). The two annotators reached consensus on all sentences for which they disagreed, resulting in a gold standard dataset with 272 (54%) subjective sentences and 232 (46%) objective sentences. More details about this dataset are available in (Mihalcea et al., 2007).

In order to learn from our annotated data, we experiment with two different classifiers, Naïve Bayes and support vector machines (SVM), selected for their performance and diversity of learning methodology. For Naïve Bayes, we use the multinomial model (McCallum and Nigam, 1998) with a threshold of 0.3. For SVM (Joachims, 1998), we use the LibSVM implementation (Fan et al., 2005) with a linear kernel.

The automatic translation of the MPQA and of the SemCor corpus was performed using Language Weaver,[1] a commercial statistical machine translation software. The resulting text was post-processed by removing diacritics, stopwords and numbers. For training, we experimented with a series of weighting schemes, yet we only report the results obtained for binary weighting, as it had the most consistent behavior.

The results obtained by running the three experiments on Romanian are shown in Table 2. The baseline on this data set is 54.16%, represented by the percentage of sentences in the corpus that are subjective, and the upper bound (UB) is 71.83%, which is the accuracy obtained under the scenario where the test data is translated into the source language and then annotated using the high-coverage OpinionFinder tool.

Perhaps not surprisingly, the SVM classifier outperforms Naïve Bayes by 2% to 6%, implying that SVM may be better fitted to lessen the amount of noise embedded in the dataset and provide more accurate classifications.

The first experiment, involving the automatic translation of the MPQA corpus enhanced with manual annotations for subjectivity at sentence level, does not seem to perform well when compared to the experiments in which automatic subjectivity classification is used. This could imply that a classifier cannot be so easily trained on the cues that humans use to express subjectivity, especially when they are not overtly expressed in the sentence and thus can be lost in the translation. Instead, the automatic annotations produced with a rule-based tool (OpinionFinder), relying on overt mentions of words in a subjectivity lexicon, seems to be more robust to translation, further resulting in better classification results. To exemplify, consider the following subjective sentence from the MPQA corpus, which does not include overt clues of subjectivity, but was annotated as subjective by the human judges because of the structure of the sentence: *It is the Palestinians that are calling for the implementation of the agreements, understandings, and recommendations pertaining to the Palestinian-Israeli conflict.*

Romanian				
Exp	Classifier	P	R	F
E1	Naïve Bayes	60.91	60.91	60.91
	SVM	66.07	66.07	66.07
E2	Naïve Bayes	63.69	63.69	63.69
	SVM	69.44	69.44	69.44
E3	Naïve Bayes	65.87	65.87	65.87
	SVM	67.86	67.86	67.86
UB	OpinionFinder	71.83	71.83	71.83

Table 2: Precision (P), Recall (R) and F-measure (F) for Romanian experiments

We compare our results with those obtained by a previously proposed method that was based on the manual translation of the SemCor subjectivity-annotated corpus. In (Mihalcea et al., 2007), we used the manual translation of the SemCor corpus into Romanian to form an English-Romanian parallel data set. The English side was annotated using the Opinion Finder tool, and the subjectivity labels were projected on the Romanian text. A Naïve Bayes classifier was then trained on the subjectivity annotated Romanian corpus and tested on the same gold standard as used in our experiments. Table 3 shows the results obtained in those experiments by using the high-coverage OpinionFinder classifier.

Among our experiments, experiments two and three are closest to those proposed in (Mihalcea et al., 2007). By using machine translation, from

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

OpinionFinder classifier	P	R	F
high-coverage	67.85	67.85	67.85

Table 3: Precision (P), Recall (R) and F-measure (F) for subjectivity analysis in Romanian obtained by using an English-Romanian parallel corpus

English into Romanian (experiment two) or Romanian into English (experiment three), and annotating this dataset with the high-coverage OpinionFinder classifier, we obtain an F-measure of 63.69%, and 65.87% respectively, using Naïve Bayes (the same machine learning classifier as used in (Mihalcea et al., 2007)). This implies that at most 4% in F-measure can be gained by using a parallel corpus as compared to an automatically translated corpus, further suggesting that machine translation is a viable alternative to devising subjectivity classification in a target language leveraged on the tools existent in a source language.

As English is a language with fewer inflections when compared to Romanian, which accommodates for gender and case as a suffix to the base form of a word, the automatic translation into English is closer to a human translation (experiment three). Therefore labeling this data using the OpinionFinder tool and projecting the labels onto a fully inflected human-generated Romanian text provides more accurate classification results, as compared to a setup where the training is carried out on machine-translated Romanian text (experiment two).

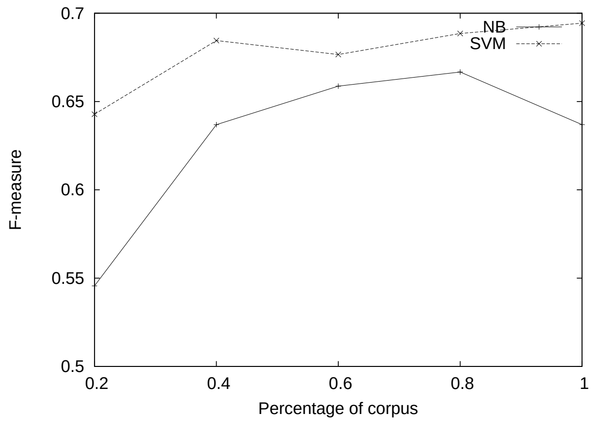


Figure 5: Experiment two: Machine learning F-measure over an incrementally larger training set

We also wanted to explore the impact that the corpus size may have on the accuracy of the classifiers. We re-ran experiments two and three with 20% corpus size increments at a time (Figures 5 and 6). It is interesting to note that a corpus of approximately 6000 sentences is able to achieve a high enough F-measure (around 66% for both experiments) to be considered viable for training a subjectivity classifier. Also, at a corpus size over 10,000 sentences, the Naïve Bayes classifier performs worse than SVM, which displays a directly proportional trend between the number of sentences in the data set and the observed F-measure. This trend could be explained by the fact that the SVM classifier is more robust with regard to noisy data, when compared to Naïve Bayes.

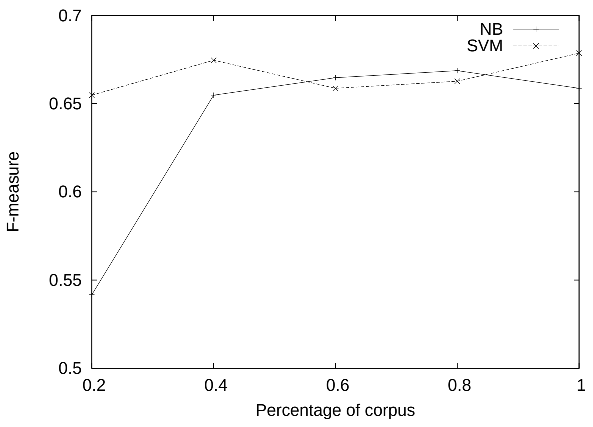


Figure 6: Experiment three: Machine learning F-measure over an incrementally larger training set

5 Portability to Other Languages

To test the validity of the results on other languages, we ran a portability experiment on Spanish.

To build a test dataset, a native speaker of Spanish translated the gold standard of 504 sentences into Spanish. We maintain the same subjectivity annotations as for the Romanian dataset. To create the training data required by the first two experiments, we translate both the MPQA corpus and the SemCor corpus into Spanish using the Google Translation service,[2] a publicly available machine translation engine also based on statistical machine translation. We were therefore able to implement all the experiments but the third, which would have required

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

a manually translated version of the SemCor corpus. Although we could have used a Spanish text to carry out a similar experiment, due to the fact that the dataset would have been different, the results would not have been directly comparable.

The results of the two experiments exploring the portability to Spanish are shown in Table 4. Interestingly, all the figures are higher than those obtained for Romanian. We assume this occurs because Spanish is one of the six official United Nations languages, and the Google translation engine is using the United Nations parallel corpus to train their translation engine, therefore implying that a better quality translation is achieved as compared to the one available for Romanian. We can therefore conclude that the more accurate the translation engine, the more accurately the subjective content is translated, and therefore the better the results. As it was the case for Romanian, the SVM classifier produces the best results, with absolute improvements over the Naïve Bayes classifier ranging from 0.2% to 3.5%.

Since the Spanish automatic translation seems to be closer to a human-quality translation, we are not surprised that this time the first experiment is able to generate a more accurate training corpus as compared to the second experiment. The MPQA corpus, since it is manually annotated and of better quality, has a higher chance of generating a more reliable data set in the target language. As in the experiments on Romanian, when performing automatic translation of the test data, we obtain the best results with an F-measure of 73.41%, which is also the upper bound on our proposed experiments.

Spanish				
Exp	Classifier	P	R	F
E1	Naïve Bayes	65.28	65.28	65.28
	SVM	68.85	68.85	68.85
E2	Naïve Bayes	62.50	62.50	62.50
	SVM	62.70	62.70	62.70
UB	OpinionFinder	73.41	73.41	73.41

Table 4: Precision (P), Recall (R) and F-measure (F) for Spanish experiments

6 Discussion

Based on our experiments, we can conclude that machine translation offers a viable approach to generating resources for subjectivity annotation in a given target language. The results suggest that either a manually annotated dataset or an automatically annotated one can provide sufficient leverage towards building a tool for subjectivity analysis.

Since the use of parallel corpora (Mihalcea et al., 2007) requires a large amount of manual labor, one of the reasons behind our experiments was to asses the ability of machine translation to transfer subjective content into a target language with minimal effort. As demonstrated by our experiments, machine translation offers a viable alternative in the construction of resources and tools for subjectivity classification in a new target language, with only a small decrease in performance as compared to the case when a parallel corpus is available and used.

To gain further insights, two additional experiments were performed. First, we tried to isolate the role played by the quality of the subjectivity annotations in the source-language for the cross-lingual projections of subjectivity. To this end, we used the high-precision OpinionFinder classifier to annotate the English datasets. As shown in Table 1, this classifier has higher precision but lower recall as compared to the high-coverage classifier we used in our previous experiments. We re-ran the second experiment, this time trained on the 3,700 sentences that were classified by the OpinionFinder high-precision classifier as either subjective or objective. For Romanian, we obtained an F-measure of 69.05%, while for Spanish we obtained an F-measure of 66.47%.

Second, we tried to isolate the role played by language-specific clues of subjectivity. To this end, we decided to set up an experiment which, by comparison, can suggest the degree to which the languages are able to accommodate specific markers for subjectivity. First, we trained an English classifier using the SemCor training data automatically annotated for subjectivity with the OpinionFinder high-coverage tool. The classifier was then applied to the English version of the manually labeled test data set (the gold standard described in Section 4). Next, we ran a similar experiment on Romanian, using a classifier trained on the Romanian version of the same

SemCor training data set, annotated with subjectivity labels projected from English. The classifier was tested on the same gold standard data set. Thus, the two classifiers used the same training data, the same test data, and the same subjectivity annotations, the only difference being the language used (English or Romanian).

The results for these experiments are compiled in Table 5. Interestingly, the experiment conducted on Romanian shows an improvement of 3.5% to 9.5% over the results obtained on English, which indicates that subjective content may be easier to learn in Romanian versus English. The fact that Romanian verbs are inflected for mood (such as indicative, conditional, subjunctive, presumptive), enables an automatic classifier to identify additional subjective markers in text. Some moods such as conditional and presumptive entail human judgment, and therefore allow for clear subjectivity annotation. Moreover, Romanian is a highly inflected language, accommodating for forms of various words based on number, gender, case, and offering an explicit lexicalization of formality and politeness. All these features may have a cumulative effect in allowing for better classification. At the same time, English entails minimal inflection when compared to other Indo-European languages, as it lacks both gender and adjective agreement (with very few notable exceptions such as *beautiful girl* and *handsome boy*). Verb moods are composed with the aid of modals, while tenses and expressions are built with the aid of auxiliary verbs. For this reason, a machine learning algorithm may not be able to identify the same amount of information on subjective content in an English versus a Romanian text. It is also interesting to note that the labeling of the training set was performed using a subjectivity classifier developed for English, which takes into account a large, human-annotated, subjectivity lexicon also developed for English. One would have presumed that any classifier trained on this annotated text would therefore provide the best results in English. Yet, as explained earlier, this was not the case.

Exp	Classifier	P	R	F
En	Naïve Bayes	60.32	60.32	60.32
	SVM	60.32	60.32	60.32
Ro	Naïve Bayes	67.85	67.85	67.85
	SVM	69.84	69.84	69.84

Table 5: Precision (P), Recall (R) and F-measure (F) for identifying language specific information

7 Conclusion

In this paper, we explored the use of machine translation for creating resources and tools for subjectivity analysis in other languages, by leveraging on the resources available in English. We introduced and evaluated three different approaches to generate subjectivity annotated corpora in a given target language, and exemplified the technique on Romanian and Spanish.

The experiments show promising results, as they are comparable to those obtained using manually translated corpora. While the quality of the translation is a factor, machine translation offers an efficient and effective alternative in capturing the subjective semantics of a text, coming within 4% F-measure as compared to the results obtained using human translated corpora.

In the future, we plan to explore additional language-specific clues, and integrate them into the subjectivity classifiers. As shown by some of our experiments, Romanian seems to entail more subjectivity markers compared to English, and this factor motivates us to further pursue the use of language-specific clues of subjectivity.

Our experiments have generated corpora of about 20,000 sentences annotated for subjectivity in Romanian and Spanish, which are available for download at http://lit.csci.unt.edu/index.php/Downloads, along with the manually annotated data sets.

Acknowledgments

The authors are grateful to Daniel Marcu and LanguageWeaver for kindly providing access to their Romanian-English and English-Romanian machine translation engines. This work was partially supported by a National Science Foundation grant IIS-#0840608.

References

C. Ovesdotter Alm, D. Roth, and R. Sproat. 2005. Emotions from text: Machine learning for text-based emotion prediction. In *Proceedings of the Human Language Technologies Conference/Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP-2005)*, pages 347–354, Vancouver, Canada.

K. Balog, G. Mishne, and M. de Rijke. 2006. Why are they excited? identifying and explaining spikes in blog mood levels. In *Proceedings of the 11th Meeting of the European Chapter of the Association for Computational Linguistics (EACL-2006)*.

A. Esuli and F. Sebastiani. 2006. Determining term subjectivity and term orientation for opinion mining. In *Proceedings the 11th Meeting of the European Chapter of the Association for Computational Linguistics (EACL-2006)*, pages 193–200, Trento, IT.

R. Fan, P. Chen, and C. Lin. 2005. Working set selection using the second order information for training svm. *Journal of Machine Learning Research*, 6:1889–1918.

M. Hu and B. Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining 2004 (KDD 2004)*, pages 168–177, Seattle, Washington.

Y. Hu, J. Duan, X. Chen, B. Pei, and R. Lu. 2005. A new method for sentiment classification in text retrieval. In *IJCNLP*, pages 1–9.

T. Joachims. 1998. Text categorization with Support Vector Machines: learning with mny relevant features. In *Proceedings of the European Conference on Machine Learning*, pages 137–142.

H. Kanayama and T. Nasukawa. 2006. Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2006)*, pages 355–363, Sydney, Australia.

N. Kando and D. Kirk Evans, editors. 2007. *Proceedings of the Sixth NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering, and Cross-Lingual Information Access*, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan, May. National Institute of Informatics.

S.-M. Kim and E. Hovy. 2006. Identifying and analyzing judgment opinions. In *Proceedings of the Human Language Technology Conference of the NAACL*, pages 200–207, New York, New York.

N. Kobayashi, K. Inui, Y. Matsumoto, K. Tateishi, and T. Fukushima. 2004. Collecting evaluative expressions for opinion extraction. In *Proceedings of the 1st International Joint Conference on Natural Language Processing (IJCNLP-04)*.

L. Lloyd, D. Kechagias, and S. Skiena. 2005. Lydia: A system for large-scale news analysis. In *String Processing and Information Retrieval (SPIRE 2005)*.

A. McCallum and K. Nigam. 1998. A comparison of event models for Naive Bayes text classification. In *Proceedings of AAAI-98 Workshop on Learning for Text Categorization*.

R. Mihalcea, C. Banea, and J. Wiebe. 2007. Learning multilingual subjective language via cross-lingual projections. In *Proceedings of the Association for Computational Linguistics*, Prague, Czech Republic.

G. Miller, C. Leacock, T. Randee, and R. Bunker. 1993. A semantic concordance. In *Proceedings of the 3rd DARPA Workshop on Human Language Technology*, Plainsboro, New Jersey.

Y. Suzuki, H. Takamura, and M. Okumura. 2006. Application of semi-supervised learning to evaluative expression classification. In *Proceedings of the 7th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2006)*, pages 502–513, Mexico City, Mexico.

H. Takamura, T. Inui, and M. Okumura. 2006. Latent variable models for semantic orientations of phrases. In *Proceedings of the 11th Meeting of the European Chapter of the Association for Computational Linguistics (EACL 2006)*, Trento, Italy.

J. Wiebe and R. Mihalcea. 2006. Word sense and subjectivity. In *Proceedings of COLING-ACL 2006*.

J. Wiebe and E. Riloff. 2005. Creating subjective and objective sentence classifiers from unannotated texts. In *Proceedings of the 6th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2005) (invited paper)*, Mexico City, Mexico.

J. Wiebe, T. Wilson, and C. Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2-3):165–210.

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

Ranking Reader Emotions Using Pairwise Loss Minimization and Emotional Distribution Regression

Kevin Hsin-Yih Lin and **Hsin-Hsi Chen**
Department of Computer Science and Information Engineering
National Taiwan University
No. 1 Roosevelt Rd. Sec. 4, Taipei, Taiwan
{f93141, hhchen}@csie.ntu.edu.tw

Abstract

This paper presents two approaches to ranking reader emotions of documents. Past studies assign a document to a single emotion category, so their methods cannot be applied directly to the emotion ranking problem. Furthermore, whereas previous research analyzes emotions from the writer's perspective, this work examines readers' emotional states. The first approach proposed in this paper minimizes pairwise ranking errors. In the second approach, regression is used to model emotional distributions. Experiment results show that the regression method is more effective at identifying the most popular emotion, but the pairwise loss minimization method produces ranked lists of emotions that have better correlations with the correct lists.

1 Introduction

Emotion analysis is an increasingly popular research topic due to the emergence of large-scale emotion data on the web. Previous work primarily studies emotional contents of texts from the writer's perspective, where it is typically assumed that a writer expresses only a single emotion in a document. Unfortunately, this premise does not hold when analyzing a document from the reader's perspective, because readers rarely agree unanimously on the emotion that a document instills. Figure 1 illustrates this phenomenon. In the figure, readers' responses are distributed among different emotion categories. In fact, none of the emotions in Figure 1 has a majority (i.e., more than 50%) of the votes. Intuitively, it is better to provide a ranking of emotions according to their popularity rather than associating a single reader emotion with a document. As a result, current writer-emotion analysis techniques for classifying a document into a single emotion category are not suitable for analyzing reader emotions. New methods capable of ranking emotions are required.

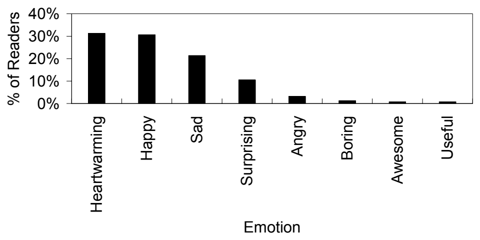


Figure 1. Emotional responses of 626 people after reading a Yahoo! News article about an Iranian refugee mother and her two children who finally reunited with their family in the March of 2007 after been stranded in a Moscow airport for 10 months due to false passports.

Reader-emotion analysis has potential applications that differ from those of writer-emotion analysis. For example, by integrating emotion ranking into information retrieval, users will be able to retrieve documents that contain relevant contents and at the same time produce desired feelings. In addition, reader-emotion analysis can assist writers in foreseeing how their work will influence readers emotionally.

Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, pages 136–144,
Honolulu, October 2008. ©2008 Association for Computational Linguistics

In this paper, we present two approaches to ranking reader emotions. The first approach is inspired by the success of the pairwise loss minimization framework used in information retrieval to rank documents. Along a similar line, we devise a novel scheme to minimize the number of incorrectly-ordered emotion pairs in a document. In the second approach, regression is used to model reader-emotion distributions directly. Experiment results show that the regression method is more effective at identifying the most popular emotion, but the pairwise loss minimization method produces ordered lists of emotions that have better correlations with the correct lists.

The rest of this paper is organized as follows. Section 2 describes related work. In Section 3, details about the two proposed approaches are provided. Section 4 introduces the corpus and Section 5 presents how features are extracted from the corpus. Section 6 shows the experiment procedures and results. Section 7 concludes the paper.

2 Related Work

Only a few studies in the past deal with the reader aspect of emotion analysis. For example, Lin et al. (2007; 2008) classify documents into reader-emotion categories. Most previous work focuses on the writer's perspective. Pang et al. (2002) design an algorithm to determine whether a document's author expresses a positive or negative sentiment. They discover that using Support Vector Machines (SVM) with word unigram features results in the best performance. Since then, more work has been done to find features better than unigrams. In (Hu et al., 2005), word sentiment information is exploited to achieve better classification accuracy.

Experiments have been done to extract emotional information from texts at granularities finer than documents. Wiebe (2000) investigates the subjectivity of words, whereas Aman and Szpakowicz (2007) manually label phrases with emotional categories. In 2007, the SemEval-2007 workshop organized a task on the unsupervised annotation of news headlines with emotions (Strapparava and Mihalcea, 2007).

As for the task of ranking, many machine-learning algorithms have been proposed in information retrieval. These techniques generate ranking functions which predict the relevance of a document. One class of algorithms minimizes the errors resulting from ordering document pairs incorrectly. Examples include (Joachims, 2002), (Freund et al., 2003) and (Qin et al., 2007). In particular, the training phase of the Joachims' Ranking SVM (Joachims, 2002) is formulated as the following SVM optimization problem:

$$\min_{\boldsymbol{w}, \xi_{i,j,k}} \tfrac{1}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} + C\sum \xi_{i,j,k}$$

subject to:

$$\begin{aligned} &\forall (q_k, d_i), (q_k, d_j) \in V \mid s_{k,i} > s_{k,j} : \\ &\quad \boldsymbol{w}^{\mathrm{T}}(\Phi(q_k, d_i) - \Phi(q_k, d_j)) \geq 1 - \xi_{i,j,k} \qquad (1) \\ &\forall i \forall j \forall k : \xi_{i,j,k} \geq 0 \end{aligned}$$

where V is the training corpus, $\Phi(q_k, d_i)$ is the feature vector of document d_i with respect to query q_k, $s_{k,i}$ is the relevance score of d_i with respect to q_k, $\boldsymbol{w}$ is a weight vector, C is the SVM cost parameter, and $\xi_{i,j,k}$ are slack variables. The set of constraints at (1) means that document pairwise orders should be preserved.

Unfortunately, the above scheme for exploiting pairwise order information cannot be applied directly to the emotion ranking task, because the task requires us to rank emotions within a document rather than provide a ranking of documents. In particular, the definitions of $\Phi(q_k,d_i)$, $\Phi(q_k,d_j)$, $s_{k,i}$ and $s_{k,j}$ do not apply to emotion ranking. In the next section, we will show how the pairwise loss minimization concept is adapted for emotion ranking.

3 Ranking Reader Emotions

In this section, we provide the formal description of the reader-emotion ranking problem. Then we describe the pairwise loss minimization (PLM) approach and the emotional distribution regression (EDR) approach to ranking emotions.

3.1 Problem Specification

The reader emotion ranking problem is defined as follows. Let $D = \{d_1, d_2, \ldots, d_N\}$ be the document space, and $E = \{e_1, e_2, \ldots, e_M\}$ be the emotion space. Let $f_i : E \rightarrow \Re$ be the emotional probability function of $d_i \in D$. That is, $f_i(e_j)$ outputs the fraction of readers who experience emotion e_j after reading document d_i. Our goal is to find a function $r : D \rightarrow E^M$ such that $r(d_i) = (e_{\pi(1)}, e_{\pi(2)}, \ldots, e_{\pi(M)})$ where π is a permutation on $\{1, 2, \ldots, M\}$, and $f_i(e_{\pi(1)}) \geq f_i(e_{\pi(2)}) \geq \ldots \geq f_i(e_{\pi(M)})$.

3.2 Pairwise Loss Minimization

As explained in Section 2, the information retrieval framework for exploiting pairwise order information cannot be applied directly to the emotion ranking problem. Hence, we introduce a novel formulation of the emotion ranking problem into an SVM optimization problem with constraints based on pairwise loss minimization.

Whereas Ranking SVM generates only a single ranking function, our method creates a pairwise ranking function $g_{jk} : D \rightarrow \Re$ for each pair of emotions e_j and e_k, aiming at satisfying the maximum number of the inequalities:

$$\forall d_i \in D \mid f_i(e_j) > f_i(e_k) : g_{jk}(d_i) > 0$$
$$\forall d_i \in D \mid f_i(e_j) < f_i(e_k) : g_{jk}(d_i) < 0$$

In other words, we want to minimize the number of incorrectly-ordered emotion pairs. We further require $g_{jk}(d_i)$ to have the linear form $\boldsymbol{w}^{\mathrm{T}}\Omega(d_i) + b$, where $\boldsymbol{w}$ is a weight vector, b is a constant, and $\Omega(d_i)$ is the feature vector of d_i. Details of feature extraction will be presented in Section 5.

As Joachims (2002) points out, the above type of problem is NP-Hard. However, an approximate solution to finding g_{ik} can be obtained by solving the following SVM optimization problem:

$$\min_{w,\, b,\, \xi_i} \ \tfrac{1}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} + C\sum \xi_i$$

subject to:

$$\forall d_i \in Q \mid f_i(e_j) > f_i(e_k) : \boldsymbol{w}^{\mathrm{T}}\Omega(d_i) + b \geq 1 - \xi_i$$
$$\forall d_i \in Q \mid f_i(e_j) < f_i(e_k) : -(\boldsymbol{w}^{\mathrm{T}}\Omega(d_i) + b) \geq 1 - \xi_i$$
$$\forall i : \xi_i \geq 0$$

where C is the SVM cost parameter, ξ_i are slack variables, and Q is the training corpus. We assume each document $d_i \in Q$ is labeled with $f_i(e_j)$ for every emotion $e_j \in E$.

When formulated as an SVM optimization problem, finding g_{jk} is equivalent to training an SVM classifier for classifying a document into the e_j or e_k category. Hence, we use LIBSVM, which is an SVM implementation, to obtain the solution.[1]

[1] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

```
Input: Set of emotion ordered pairs $P$
1. $G \leftarrow$ a graph with emotions as vertices and no edge
2. while ($P \neq \emptyset$)
3.   remove $(e_j,e_k)$ with the highest confidence from $P$
4.   if adding edge $(e_j,e_k)$ to $G$ produces a loop
5.     then add $(e_k,e_j)$ to $G$
6.   else add $(e_j,e_k)$ to $G$
7. return topological sort of $G$
```

Algorithm 1. Merge Pairwise Orders.

We now describe how we rank the emotions of a previously unseen document using the $M(M-1)/2$ pairwise ranking functions g_{jk} created during the training phase. First, all of the pairwise ranking functions are applied to the unseen document, which generates the relative orders of every pair of emotions. These pairwise orders need to be combined together to produce a ranked list of all the emotions. Algorithm 1 does exactly this.

In Algorithm 1, the confidence of an emotion ordered pair at Line 3 is the probability value returned by a LIBSVM classifier for predicting the order. LIBSVM's method for generating this probability is described in (Wu et al., 2003). Lines 4 and 5 resolve the problem of conflicting emotion ordered pairs forming a loop in the ordering of emotions. The ordered list of emotions returned by Algorithm 1 at Line 7 is the final output of the PLM method.

3.3 Emotional Distribution Regression

In the second approach to ranking emotions, we use regression to model f_i directly. A regression function $h_j : D \rightarrow \Re$ is generated for each $e_j \in E$ by learning from the examples $(\Omega(d_i), f_i(e_j))$ for all documents d_i in the training corpus.

The regression framework we adopt is Support Vector Regression (SVR), which is a regression analysis technique based on SVM (Schölkopf et al., 2000). We require h_j to have the form $\boldsymbol{w}^{\mathrm{T}}\Omega(d_i) + b$. Finding h_j is equivalent to solving the following optimization problem:

$$\min_{w,\, b,\, \xi_{i,1},\, \xi_{i,2}} \ \tfrac{1}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} + C\sum(\xi_{i,1} + \xi_{i,2})$$

subject to:

$$\forall d_i \in Q :$$
$$f_i(e_j) - (\boldsymbol{w}^{\mathrm{T}}\Omega(d_i) + b) \geq \varepsilon - \xi_{i,1}$$
$$(\boldsymbol{w}^{\mathrm{T}}\Omega(d_i) + b) - f_i(e_j) \geq \varepsilon - \xi_{i,2}$$
$$\forall i : \xi_{i,1}, \xi_{i,2} \geq 0$$

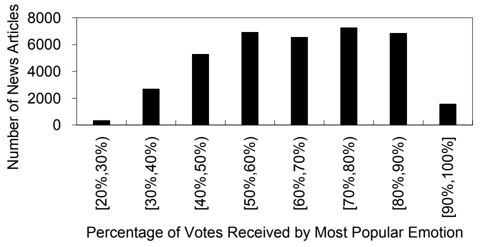


Figure 2. News articles in the entire corpus grouped by the percentage of votes received by the most popular emotion.

where C is the cost parameter, ε is the maximum difference between the predicted and actual values we wish to maintain, $\xi_{i,1}$ and $\xi_{i,2}$ are slack variables, and Q is the training corpus. To solve the above optimization problem, we use SVM^{light}'s SVR implementation.[2]

When ranking the emotions of a previously unseen document d_k, we sort the emotions $e_j \in E$ in descending order of $h_j(d_k)$.

4 Constructing the Corpus

The training and test corpora used in this study comprise Chinese news articles from Yahoo! Kimo News[3], which allows a user to cast a vote for one of eight emotions to express how a news article makes her feel. Each Yahoo! news article contains a list of eight emotions at the bottom of the webpage. A reader may select one of the emotions and click on a submit button to submit the emotion. As with many websites which collect user responses, such as the Internet Movie Database, users are not forced to submit their responses. After submitting a response, the user can view a distribution of emotions indicating how other readers feel about the same article. Figure 1 shows the voting results of a Yahoo! news article.

The eight available emotions are *happy*, *sad*, *angry*, *surprising*, *boring*, *heartwarming*, *awesome*, and *useful*. *Useful* is not a true emotion. Rather, it means that a news article contains practical information. The value $f_i(e_j)$ is derived by normalizing the number of votes for emotion e_j in document d_i by the total number votes in d_i.

The entire corpus consists of 37,416 news articles dating from January 24, 2007 to August 7, 2007. News articles prior to June 1, 2007 form the training corpus (25,975 articles), and the remaining ones form the test corpus (11,441 articles). We collect articles a week after their publication dates to ensure that the vote counts have stabilized.

As mentioned earlier, readers rarely agree unanimously on the emotion of a document. Figure 2 illustrates this. In 41% of all the news articles in the entire corpus, the most popular emotion receives less than 60% of the votes.

5 Extracting Features

After obtaining news articles, the next step is to determine how to convert them into feature vectors for SVM and SVR. That is, we want to instantiate Ω. For this purpose, three types of features are extracted.

The first feature type consists of Chinese character bigrams, which are taken from the headline and content of each news article. The presence of a bigram is indicated by a binary feature value.

Chinese words form the second type of features. Unlike English words, consecutive Chinese words in a sentence are not separated by spaces. To deal with this problem, we utilize Stanford NLP Group's Chinese word segmenter to split a sentence into words.[4] As in the case of bigrams, binary feature values are used.

We use character bigram features in addition to word features to increase the coverage of Chinese words. A Chinese word is formed by one or more contiguous Chinese characters. As mentioned earlier, Chinese words in a sentence are not separated by any boundary symbol (e.g., a space), so a Chinese word segmentation tool is always required to extract words from a sentence. However, a word segmenter may identify word boundaries erroneously, resulting in the loss of correct Chinese words. This problem is particularly severe if there are a lot of out-of-vocabulary words in a dataset. In Chinese, around 70% of all Chinese words are Chinese character bigrams (Chen et al., 1997). Thus, using Chinese character bigrams as features will allow us to identify a lot of Chinese words, which when combined with the words extracted by the word segmenter, will give us a wider coverage of Chinese words.

The third feature type is extracted from news metadata. A news article's metadata are its news

[2] http://svmlight.joachims.org/
[3] http://tw.news.yahoo.com

[4] http://nlp.stanford.edu/software/segmenter.shtml

category, agency, hour of publication, reporter, and event location. Examples of news categories include sports and political. Again, we use binary feature values. News metadata are used because they may contain implicit emotional information.

6 Experiments

The experiments are designed to achieve the following four goals: (i) to compare the ranking performance of different methods, (ii) to analyze the pairwise ranking quality of PLM, (iii) to analyze the distribution estimation quality of EDR, and (iv) to compare the ranking performance of different feature sets. The Yahoo! News training and test corpora presented in Section 4 are used in all experiments.

6.1 Evaluation Metrics for Ranking

We employ three metrics as indicators of ranking quality: ACC@*k*, NDCG@*k* and SACC@*k*.

ACC@*k* stands for accuracy at position *k*. According to ACC@*k*, a predicted ranked list is correct if the list's first *k* items are identical (i.e., same items in the same order) to the true ranked list's first *k* items. If two emotions in a list have the same number of votes, then their positions are interchangeable. ACC@*k* is computed by dividing the number of correctly-predicted instances by the total number of instances.

NDCG@*k*, or normalized discounted cumulative gain at position *k* (Järvelin and Kekäläinen, 2002), is a metric frequently used in information retrieval to judge the quality of a ranked list when multiple levels of relevance are considered. This metric is defined as

$$\text{NDCG}@k = z_k \sum_{i=1}^{k} \frac{rel_i}{\log_2(i+1)}$$

where rel_i is the relevance score of the predicted item at position *i*, and z_k is a normalizing factor which ensures that a correct ranked list has an NDCG@*k* value of 1. In the emotion ranking problem, rel_i is the percentage of reader votes received by the emotion at position *i*. Note that the $\log_2(i+1)$ value in the denominator is a discount factor which decreases the weights of items ranked later in a list. NDCG@*k* has the range [0, 1], where 1 is the best. In the experiment results, NDCG@*k* values are averaged over all instances in the test corpus.

NDCG@*k* is used because ACC@*k* has the disadvantage of not taking emotional distributions into account. Take Figure 1 as an example. In the figure, *heartwarming* and *happy* have 31.3% and 30.7% of the votes, respectively. Since the two percentages are very close, it is reasonable to say that predicting *happy* as the first item in a ranked list may also be acceptable. However, doing so would be completely incorrect according to ACC@*k*. In contrast, NDCG@*k* would consider it to be partially correct, and the extent of correctness depends on how much *heartwarming* and *happy*'s percentages of votes differ. To be exact, if *happy* is predicted as the first item, then the corresponding NDCG@1 would be 30.7% / 31.3% = 0.98.

The third metric is SACC@*k*, or set accuracy at *k*. It is a variant of ACC@*k*. According to SACC@*k*, a predicted ranked list is correct if the set of its first *k* items is the same as the true ranked list's set of first *k* items. In effect, SACC@*k* evaluates a ranking method's ability to place the top *k* most important items in the first *k* positions.

6.2 Tuning SVM and SVR Parameters

SVM and SVR are employed in PLM and EDR, respectively. Both SVM and SVR have the adjustable *C* cost parameter, and SVR has an additional ε parameter. To estimate the optimal *C* value for a combination of SVM and features, we perform 4-fold cross-validation on the Yahoo! News training corpus, and select the *C* value which results in the highest binary classification accuracy during cross-validation. The same procedure is used to estimate the best *C* and ε values for a combination of SVR and features. The C-ε pair which results in the lowest mean squared error during cross-validation is chosen. The candidate *C* values for both SVM and SVR are 2^{-10}, 2^{-9}, …, 2^{-6}. The candidate ε values for SVR are 10^{-2} and 10^{-1}. All cross-validations are performed solely on the training data. The test data are not used to tune the parameters. Also, SVM and SVR allow users to specify the type of kernel to use. Linear kernel is selected for both SVM and SVR.

6.3 Nearest Neighbor Baseline

The nearest neighbor (NN) method is used as the baseline. The ranked emotion list of a news article in the test corpus is predicted as follows. First, the

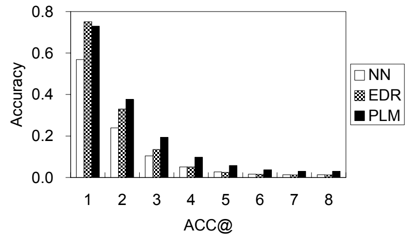


Figure 3. ACC@k

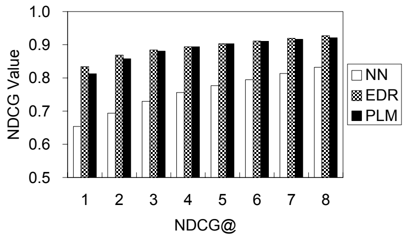


Figure 4. NDCG@k

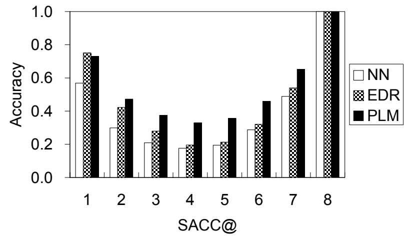


Figure 5. SACC@k

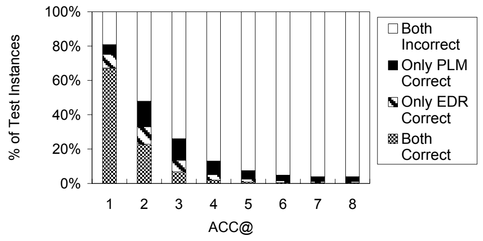


Figure 6. Performance of PLM and EDR.

test news article is compared to every training news article using cosine similarity, which is defined as

$$\cos(d_i, d_j) = \frac{|D_i \cap D_j|}{\sqrt{|D_i| \times |D_i|}}$$

where d_i and d_j are two news articles, and D_i and D_j are sets of Chinese character bigrams in d_i and d_j, respectively. The ranked emotion list of the training article having the highest cosine similarity with the test article is used as the predicted ranked list.

6.4 Comparison of Methods

Figures 3 to 5 show the performance of different ranking methods on the test corpus. For both PLM and EDR, all of the bigram, word, and news metadata features are used.

In Figure 3, EDR's ACC@1 (0.751) is higher than those of PLM and NN, and the differences are statistically significant with p-value < 0.01. So, EDR is the best method at predicting the most popular emotion. However, PLM has the best ACC@k for $k \geq 2$, and the differences from the other two methods are all significant with p-value < 0.01. This means that PLM's predicted ranked lists better resemble the true ranked lists.

Figure 3 displays a sharp decrease in ACC@k values as k increases. This trend indicates the hardness of predicting a ranked list correctly. Looking from a different angle, the ranking task under the ACC@k metric is equivalent to the classification of news articles into one of $8!/(8-k)!$ classes, where we regard each unique emotion sequence of length k as a class. In fact, computing ACC@8 for a ranking method is the same as evaluating the method's ability to classify a news article into one of 8! = 40,320 classes. So, producing a completely-correct ranked list is a difficult task.

In Figure 4, all of PLM and EDR's NDCG@k improvements over NN are statistically significant with p-value < 0.01. For some values of k, the difference in NDCG@k between PLM and EDR is not significant. The high NDCG@k values (i.e., greater than 0.8) of PLM and EDR imply that although it is difficult for PLM and EDR to generate completely-correct ranked lists, these two methods are effective at placing highly popular emotions to the beginning of ranked lists.

In Figure 5, PLM outperforms the other two methods for $2 \leq k \leq 7$, and the differences are all statistically significant with p-value < 0.01. For small values of k (e.g., $2 \leq k \leq 3$), PLM's higher SACC@k values mean that PLM is better at placing the highly popular emotions in the top positions of a ranked list.

To further compare PLM and EDR, we examine their performance on individual test instances. Figure 6 shows the percentage of test instances where both PLM and EDR give incorrect lists, only PLM gives correct lists, only EDR gives ranked lists, and both methods give correct lists. The "Only PLM Correct" and "Only EDR Correct" categories are nonzero, so neither PLM nor EDR is always better than the other.

In summary, EDR is the best at predicting the most popular emotion according to ACC@1, NDCG@1 and SACC@1. However, PLM generates ranked lists that better resemble the correct ranked lists according to ACC@k and SACC@k

Method	Average τ_b	Average p-value
PLM	0.584	0.068
EDR	0.474	0.114
NN	0.392	0.155

Table 1. Kendall's τ_b statistics.

	He	Su	Sa	Us	Ha	Bo	An
Aw	0.80	0.75	0.78	0.77	0.82	0.76	0.79
He		0.79	0.81	0.78	0.81	0.89	0.81
Su			0.82	0.78	0.80	0.82	0.82
Sa				0.78	0.80	0.84	0.82
Us					0.82	0.91	0.82
Ha						0.83	0.79
Bo							0.80

Table 2. Classification accuracies of SVM pairwise emotion classifiers on the test corpus. He = *heartwarming*, Su = *surprising*, Sa = *sad*, Us = *useful*, Ha = *happy*, Bo = *boring*, and An = *angry*.

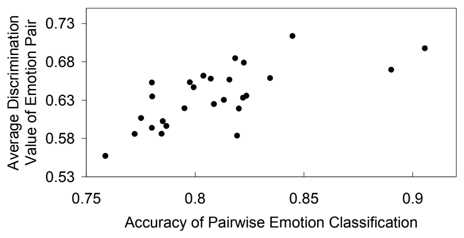


Figure 7. Accuracy of pairwise emotion classification and the corresponding average discrimination value.

for $k \geq 2$. Further analysis shows that neither method is always better than the other.

6.5 Pairwise Ranking Quality of PLM

In this subsection, we evaluate the performance of PLM in predicting pairwise orders.

We first examine the quality of ranked lists generated by PLM in terms of pairwise orders. To do this, we use Kendall's τ_b correlation coefficient, which is a statistical measure for determining the correlation between two ranked lists when there may be ties between two items in a list (Liebetrau, 1983). The value of τ_b is determined based on the number of concordant pairwise orders and the number of discordant pairwise orders between two ranked lists. Therefore, this measure is appropriate for evaluating the effectiveness of PLM at predicting pairwise orders correctly. τ_b has the range [-1, 1], where 1 means a perfect positive correlation, and -1 means two lists are the reverse of each other. When computing τ_b of two ranked lists, we also calculate a p-value to indicate whether the correlation is statistically significant.

We compute τ_b statistics between a predicted ranked list and the corresponding true ranked list. Table 1 shows the results. In Table 1, numbers in the "Average τ_b" and "Average p-value" columns are averaged over all test instances. The statistics for EDR and NN are also included for comparison. From the table, we see that PLM has the highest average τ_b value and the lowest average p-value, so PLM is better at preserving pairwise orders than EDR and NN methods. This observation verifies that PLM's minimization of pairwise loss leads to better prediction of pairwise orders.

We now look at the individual performance of the 28 pairwise emotion rankers g_{jk}. As mentioned in Section 3.2, each pairwise emotion ranker g_{jk} is equivalent to a binary classifier for classifying a document into the e_j or e_k category. So, we look at their classification accuracies in Table 2. In the table, accuracy ranges from 0.75 for the *awesome-surprising* pair to 0.91 for the *useful-boring* pair.

From the psychological perspective, the relatively low accuracy of the *awesome-surprising* pair is expected, because *awesome* is *surprising* in a positive sense. So, readers should have a hard time distinguishing between these two emotions. And the SVM classifier, which models reader responses, should also find it difficult to discern these two emotions. Based on this observation, we suspect that the pairwise classification performance actually reflects the underlying emotional ambiguity experienced by readers. To verify this, we quantify the degree of ambiguity between two emotions, and compare the result to pairwise classification accuracy.

To quantify emotional ambiguity, we introduce the concept of discrimination value between two emotions e_j and e_k in a document d_i, which is defined as follows:

$$\frac{\left| f_i(e_j) - f_i(e_k) \right|}{f_i(e_j) + f_i(e_k)}$$

where f_i is the emotional probability function defined in Section 3.1. Intuitively, the larger the discrimination value is, the smaller the degree of ambiguity between two emotions is.

Figure 7 shows the relationship between pairwise classification accuracy and the average discrimination value of the corresponding emotion

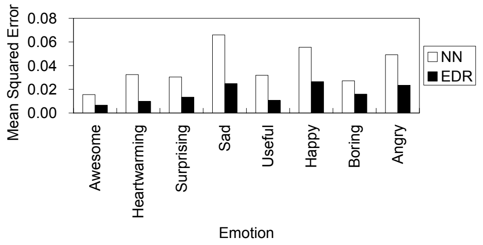


Figure 8. Mean squared error of NN and EDR for estimating the emotional distributions of the test corpus.

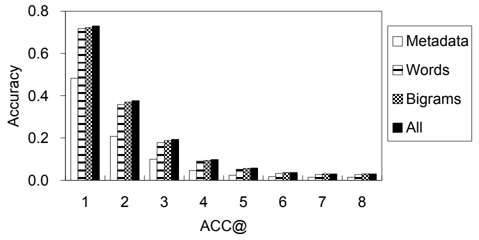


Figure 9. PLM performance using different features.

pair. The general pattern is that as accuracy increases, the discrimination value also increases. To provide concrete evidence, we use Pearson's product-moment correlation coefficient, which has the range of [-1, 1], where 1 means a perfect positive correlation (Moore, 2006). The coefficient for the data in Figure 7 is 0.726 with p-value < 0.01. Thus, pairwise emotion classification accuracy reflects the emotional ambiguity experienced by readers.

In summary, PLM's pairwise loss minimization leads to better pairwise order predictions than EDR and NN. Also, the pairwise classification results reveal the inherent ambiguity between emotions.

6.6 Distribution Estimation Quality of EDR

In this subsection, we evaluate EDR's performance in estimating the emotional probability function f_i.

With the prior knowledge that a news article's f_i values sum to 1 over all emotions, and f_i is between 0 and 1, we adjust EDR's f_i predictions to produce proper distributions. It is done as follows. A predicted f_i value greater than 1 or less than 0 is set to 1 and 0, respectively. Then the predicted f_i values are normalized to sum to 1 over all emotions.

NN's distribution estimation performance is included for comparison. For NN, the predicted f_i values of a test article are taken from the emotional distribution of the most similar training article.

Figure 8 shows the mean squared error of EDR and NN for predicting f_i. In the figure, the error generated by EDR is less than those by NN, and all the differences are statistically significant with p-value < 0.01. Thus, EDR's use of regression leads to better estimation of f_i than the NN.

6.7 Comparison of Features

Figure 9 shows each of the three feature type's ACC@k for predicting test instances' ranked lists when PLM is used. The feature comparison graph for EDR is not shown, because it exhibits a very similar trend as PLM. For both PLM and EDR, bigrams are better than words, which are in turn better than news metadata. In Figure 9, the combination of all three feature sets achieves the best performance. For both PLM and EDR, the improvements in ACC@k of using all features over words and metadata are all significant with p-value < 0.01, and the improvements over bigrams are significant for $k \leq 2$. Hence, in general, it is better to use all three feature types together.

7 Conclusions and Future Work

This paper presents two methods to ranking reader emotions. The PLM method minimizes pairwise loss, and the EDR method estimates emotional distribution through regression. Experiments with significant tests show that EDR is better at predicting the most popular emotion, but PLM produces ranked lists that have higher correlation with the correct lists. We further verify that PLM has better pairwise ranking performance than the other two methods, and EDR has better distribution estimation performance than NN.

As for future work, there are several directions we can pursue. An observation is that PLM exploits pairwise order information, whereas EDR exploits emotional distribution information. We plan to combine these two methods together. Another research direction is to improve EDR by finding better features. We would also like to integrate emotion ranking into information retrieval.

Acknowledgments

We are grateful to the Computer and Information Networking Center, National Taiwan University, for the support of high-performance computing facilities. The research in this paper was partially supported by National Science Council, Taiwan, under the contract NSC 96-2628-E-002-240-MY3.

References

Saima Aman and Stan Szpakowicz. 2007. *Identifying Expressions of Emotion in Text*. In Proceedings of 10th International Conference on Text, Speech and Dialogue, Lecture Notes in Computer Science 4629, 196-205. Springer, Plzeň, CZ.

Aitao Chen, Jianzhang He, Liangjie Xu, Frederic Gey, and Jason Meggs. 1997. *Chinese Text Retrieval wihtout using a Dictionary.* In Proceedings of 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 42-49. Association for Computing Machinery, Philadelphia, US.

Yoav Freund, Raj D. Iyer, Robert E. Schapire, and Yoram Singer. 2003. *An Efficient Boosting Algorithm for Combining Preferences*. Journal of Machine Learning Research, 4, 933-969.

Yi Hu, Jianyong Duan, Xiaoming Chen, Bingzhen Pei, and Ruzhan Lu. 2005. *A New Method for Sentiment Classification in Text Retrieval*. In Proceedings of 2nd International Joint Conference on Natural Language Processing, 1-9. Jeju Island, KR.

Kalervo Järvelin and Jaana Kekäläinen. *Cumulative Gain-based Evaluation of IR Techniques*. 2002. ACM Transactions on Information Systems, 20(4), 422-446.

Thorsten Joachims. 2002. *Optimizing Search Engines using Clickthrough Data*. In Proceedings of 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Association for Computing Machinery, Edmonton, CA.

Albert M. Liebetrau. 1983. *Measures of Association*. Sage Publications, Newbury Park, US.

Kevin H. Lin, Changhua Yang, and Hsin-Hsi Chen. 2007. *What Emotions do News Articles Trigger in their Readers?* In Proceedings of 30th ACM SIGIR Conference, 733-734. Association for Computing Machinery, Amsterdam, NL.

Kevin H. Lin, Changhua Yang, and Hsin-Hsi Chen. 2008. *Emotion Classification of Online News Articles from the Reader's Perspective*. In Proceedings of International Conference on Web Intelligence. Institute of Electrical and Electronics Engineers, Sydney, AU.

David Moore. 2006. *The Basic Practice of Statistics*. W.H. Freeman and Company, New York, US.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. *Thumbs up? Sentiment Classification Using Machine Learning Techniques*. In Proceedings of 2002 Conference on Empirical Methods in Natural Language Processing, 79-86. Association for Computational Linguistics, Philadelphia, US.

Tao Qin, Tie-Yan Liu, Wei Lai, Xu-Dong Zhang, De-Sheng Wang, and Hang Li. 2007. *Ranking with Multiple Hyperplanes*. In Proceedings of 30th ACM SIGIR Conference, 279-286. Association for Computing Machinery, Amsterdam, NL.

Bernhard Schölkopf, Alex J. Smola, Robert C. Williamson, and Peter L. Barlett. 2000. *New Support Vector Algorithms*. Neural Computation, 12(5), 1207-1245.

Carlo Strapparava and Rada Mihalcea. 2007. *SemEval-2007 Task 14: Affective Text*. In Proceedings of 4th International Workshop on Semantic Evaluations. Prague, CZ.

Janyce M. Wiebe. 2000. *Learning Subjective Adjectives from Corpora*. In Proceedings of 17th Conference of the American Association for Artificial Intelligence, 735-740. AAAI Press, Austin, US.

Ting-Fan Wu, Chih-Jen Lin, and Ruby C. Weng. *Probability Estimates for Multi-class Classification by Pairwise Coupling*. 2004. Journal of Machine Learning Research, 5, 975-1005.

Dependency Parsing by Belief Propagation*

David A. Smith and **Jason Eisner**
Dept. of Computer Science, Johns Hopkins University
Balitmore, MD 21218, USA
{dasmith,eisner}@jhu.edu

Abstract

We formulate dependency parsing as a graphical model with the novel ingredient of global constraints. We show how to apply loopy belief propagation (BP), a simple and effective tool for *approximate* learning and inference. As a parsing algorithm, BP is both asymptotically and empirically efficient. Even with second-order features or latent variables, which would make exact parsing considerably slower or NP-hard, BP needs only $O(n^3)$ time with a small constant factor. Furthermore, such features significantly improve parse accuracy over exact first-order methods. Incorporating additional features would increase the runtime additively rather than multiplicatively.

1 Introduction

Computational linguists worry constantly about runtime. Sometimes we oversimplify our models, trading linguistic nuance for fast dynamic programming. Alternatively, we write down a better but intractable model and then use approximations. The CL community has often approximated using heavy pruning or reranking, but is beginning to adopt other methods from the machine learning community, such as Gibbs sampling, rejection sampling, and certain variational approximations.

We propose borrowing a different approximation technique from machine learning, namely, loopy belief propagation (BP). In this paper, we show that BP can be used to *train* and *decode* complex parsing models. Our approach calls a simpler parser as a subroutine, so it still exploits the useful, well-studied combinatorial structure of the parsing problem.[1]

2 Overview and Related Work

We wish to make a dependency parse's score depend on *higher-order* features, which consider arbitrary interactions among two or more edges in the parse (and perhaps also other latent variables such as part-of-speech tags or edge labels). Such features can help accuracy—as we show. Alas, they raise the polynomial runtime of projective parsing, and render non-projective parsing NP-hard. Hence we seek approximations.

We will show how BP's "message-passing" discipline offers a principled way for *higher-order* features to incrementally adjust the numerical edge weights that are fed to a fast *first-order* parser. Thus the first-order parser is influenced by higher-order interactions among edges—but not asymptotically slowed down by considering the interactions itself.

BP's behavior in our setup can be understood intuitively as follows. Inasmuch as the first-order parser finds that edge e is probable, the higher-order features will kick in and discourage other edges e' to the extent that they prefer not to coexist with e.[2] Thus, the *next* call to the first-order parser assigns lower probabilities to parses that contain these e'. (The method is approximate because a first-order parser must equally penalize *all* parses containing e', even those that do not in fact contain e.)

This behavior is somewhat similar to parser stacking (Nivre and McDonald, 2008; Martins et al., 2008), in which a first-order parser derives some of its input features from the full 1-best output of another parser. In our method, a first-order parser derives such input features from its *own* previous full output (but probabilistic output rather than just 1-best). This circular process is *iterated* to convergence. Our method also permits the parse to interact cheaply with other variables. Thus first-order parsing, part-of-speech tagging, and other tasks on a common input could mutually influence one another.

Our method and its numerical details emerge naturally as an instance of the well-studied loopy BP algorithm, suggesting several potential future im-

*This work was supported by the Human Language Technology Center of Excellence.

[1] As do constraint relaxation (Tromble and Eisner, 2006) and forest reranking (Huang, 2008). In contrast, generic NP-hard solution techniques like Integer Linear Programming (Riedel and Clarke, 2006) know nothing about optimal substructure.

[2] This may be reminiscent of adjusting a Lagrange multiplier on e' until some (hard) constraint is satisfied.

provements to accuracy (Yedidia et al., 2004; Braunstein et al., 2005) and efficiency (Sutton and McCallum, 2007).

Loopy BP has occasionally been used before in NLP, with good results, to handle non-local features (Sutton and McCallum, 2004) or joint decoding (Sutton et al., 2004). However, our application to parsing requires an innovation to BP that we explain in §5—a *global* constraint to enforce that the parse is a tree. The tractability of some such global constraints points the way toward applying BP to other computationally intensive NLP problems, such as syntax-based alignment of parallel text.

3 Graphical Models of Dependency Trees

3.1 Observed and hidden variables

To apply BP, we must formulate dependency parsing as a search for an optimal **assignment** to the variables of a graphical model. We encode a parse using the following variables:

Sentence. The n-word input sentence W is fully observed (not a lattice). Let $W = W_0 W_1 \cdots W_n$, where W_0 is always the special symbol ROOT.

Tags. If desired, the variables $T = T_1 T_2 \cdots T_n$ may specify tags on the n words, drawn from some tagset $\mathcal{T}$ (e.g., parts of speech). These variables are needed iff the tags are to be inferred jointly with the parse.

Links. The $O(n^2)$ boolean variables $\{L_{ij} : 0 \leq i \leq n, 1 \leq j \leq n, i \neq j\}$ correspond to the possible links in the dependency parse.[3] $L_{ij} = \mathsf{true}$ is interpreted as meaning that there exists a dependency link from parent $i \rightarrow$ child j.[4]

Link roles, etc. It would be straightforward to add other variables, such as a binary variable L_{irj} that is true iff there is a link $i \xrightarrow{r} j$ labeled with role r (e.g., AGENT, PATIENT, TEMPORAL ADJUNCT).

3.2 Markov random fields

We wish to define a probability distribution over all configurations, i.e., all joint assignments $\mathcal{A}$ to these variables. Our distribution is simply an undirected graphical model, or Markov random field (MRF):[5]

$$p(\mathcal{A}) \overset{\text{def}}{=} \frac{1}{Z} \prod_m F_m(\mathcal{A}) \tag{1}$$

specified by the collection of **factors** $F_m : \mathcal{A} \mapsto \mathbb{R}^{\geq 0}$. Each factor is a function that consults only a *subset* of $\mathcal{A}$. We say that the factor has **degree** d if it depends on the values of d variables in $\mathcal{A}$, and that it is **unary**, **binary**, **ternary**, or **global** if d is respectively 1, 2, 3, or unbounded (grows with n).

A factor function $F_m(\mathcal{A})$ may also depend freely on the observed variables—the input sentence W and a known (learned) parameter vector θ. For notational simplicity, we suppress these extra arguments when writing and drawing factor functions, and when computing their degree. In this treatment, these observed variables are not specified by $\mathcal{A}$, but instead are absorbed into the very definition of F_m.

In defining a factor F_m, we often define the circumstances under which it **fires**. These are the only circumstances that allow $F_m(\mathcal{A}) \neq 1$. When F_m does not fire, $F_m(\mathcal{A}) = 1$ and does not affect the product in equation (1).

3.3 Hard constraints

A **hard** factor F_m fires only on parses $\mathcal{A}$ that *violate* some specified condition. It has value 0 on those parses, acting as a hard constraint to rule them out.

TREE. A hard global constraint on all the L_{ij} variables at once. It requires that exactly n of these variables be true, and that the corresponding links form a directed tree rooted at position 0.

PTREE. This stronger version of TREE requires further that the tree be **projective**. That is, it prohibits L_{ij} and $L_{k\ell}$ from both being true if $i \rightarrow j$ crosses $k \rightarrow \ell$. (These links are said to **cross** if one of k, ℓ is strictly between i and j while the other is strictly outside that range.)

EXACTLY1. A *family* of $O(n)$ hard global constraints, indexed by $1 \leq j \leq n$. EXACTLY1$_j$ requires that j have exactly one parent, i.e., exactly one of the L_{ij} variables must be true. Note that EXACTLY1 is implied by TREE or PTREE.

[3]"Links" are conventionally called edges, but we reserve the term "edge" for describing the graphical model's factor graph.

[4]We could have chosen a different representation with $O(n)$ integer variables $\{P_j : 1 \leq j \leq n\}$, writing $P_j = i$ instead of $L_{ij} = \mathsf{true}$. This representation can achieve the same asymptotic runtime for BP by using sparse messages, but some constraints and algorithms would be somewhat harder to explain.

[5]Our overall model is properly called a *dynamic* MRF, since we must construct different-size MRFs for input sentences of different lengths. Parameters are shared both across and within these MRFs, so that only finitely many parameters are needed.

ATMOST1. A weaker version. ATMOST1_j requires j to have one or zero parents.

NAND. A *family* of hard binary constraints. NAND$_{ij,k\ell}$ requires that L_{ij} and $L_{k\ell}$ may not both be true. We will be interested in certain subfamilies.

NOT2. Shorthand for the family of $O(n^3)$ binary constraints $\{\text{NAND}_{ij,kj}\}$. These are collectively equivalent to ATMOST1, but expressed via a larger number of simpler constraints, which can make the BP approximation less effective (footnote 30).

NO2CYCLE. Shorthand for the family of $O(n^2)$ binary constraints $\{\text{NAND}_{ij,ji}\}$.

3.4 Soft constraints

A **soft** factor F_m acts as a soft constraint that prefers some parses to others. In our experiments, it is always a log-linear function returning positive values:

$$F_m(\mathcal{A}) \overset{\text{def}}{=} \exp \sum_{h \in \text{features}(F_m)} \theta_h f_h(\mathcal{A}, W, m) \quad (2)$$

where θ is a learned, finite collection of weights and f is a corresponding collection of feature functions, some of which are used by F_m. (Note that f_h is permitted to consult the observed input W. It also sees which factor F_m it is scoring, to support reuse of a single feature function f_h and its weight θ_h by unboundedly many factors in a model.)

LINK. A family of unary soft factors that judge the links in a parse $\mathcal{A}$ individually. LINK$_{ij}$ fires iff L_{ij} = true, and then its value depends on (i, j), W, and θ. Our experiments use the same features as McDonald et al. (2005).

A first-order (or "edge-factored") parsing model (McDonald et al., 2005) contains *only* LINK factors, along with a global TREE or PTREE factor. Though there are $O(n^2)$ link factors (one per L_{ij}), only n of them fire on any particular parse, since the global factor ensures that exactly n are true.

We'll consider various higher-order soft factors:

PAIR. The binary factor PAIR$_{ij,k\ell}$ fires with some value iff L_{ij} and $L_{k\ell}$ are both true. Thus, it penalizes or rewards a pair of links for being simultaneously present. This is a soft version of NAND.

GRAND. Shorthand for the family of $O(n^3)$ binary factors $\{\text{PAIR}_{ij,jk}\}$, which evaluate grandparent-parent-child configurations, $i \to j \to k$. For example, whether preposition j attaches to verb i might depend on its object k. In non-projective parsing, we might prefer (but not require) that a parent and child be on the same side of the grandparent.

SIB. Shorthand for the family of $O(n^3)$ binary factors $\{\text{PAIR}_{ij,ik}\}$, which judge whether two children of the same parent are compatible. E.g., a given verb may not like to have two noun children both to its left.[6] The children do not need to be adjacent.

CHILDSEQ. A family of $O(n)$ global factors. CHILDSEQ$_i$ scores i's sequence of children; hence it consults all variables of the form L_{ij}. The scoring follows the parametrization of a weighted split head-automaton grammar (Eisner and Satta, 1999). If 5 has children $2, 7, 9$ under $\mathcal{A}$, then CHILDSEQ$_i$ is a product of subfactors of the form PAIR$_{5\#,57}$, PAIR$_{57,59}$, PAIR$_{59,5\#}$ (right child sequence) and PAIR$_{5\#,52}$, PAIR$_{52,5\#}$ (left child sequence).

NOCROSS. A family of $O(n^2)$ global constraints. If the parent-to-j link crosses the parent-to-ℓ link, then NOCROSS$_{j\ell}$ fires with a value that depends only on j and ℓ. (If j and ℓ do not each have exactly one parent, NOCROSS$_{j\ell}$ fires with value 0; i.e., it incorporates EXACTLY1_j and EXACTLY1_ℓ.)[7]

TAG$_i$ is a unary factor that evaluates whether T_i's value is consistent with W (especially W_i).

TAGLINK$_{ij}$ is a ternary version of the LINK$_{ij}$ factor whose value depends on L_{ij}, T_i and T_j (i.e., its feature functions consult the tag variables to decide whether a link is likely). One could similarly enrich the other features above to depend on tags and/or link roles; TAGLINK is just an illustrative example.

TRIGRAM is a global factor that evaluates the tag sequence T according to a trigram model. It is a product of subfactors, each of which scores a trigram of adjacent tags T_{i-2}, T_{i-1}, T_i, possibly also considering the word sequence W (as in CRFs).

4 A Sketch of Belief Propagation

MacKay (2003, chapters 16 and 26) provides an excellent introduction to belief propagation, a gen-

[6] A similar binary factor could directly discourage giving the verb two SUBJECTs, if the model has variables for link roles.

[7] In effect, we have combined the $O(n^4)$ binary factors PAIR$_{ij,k\ell}$ into $O(n^2)$ groups, and made them more precise by multiplying in EXACTLYONE constraints (see footnote 30). This will permit $O(n^3)$ total computation if we are willing to sacrifice the ability of the PAIR weights to depend on i and k.

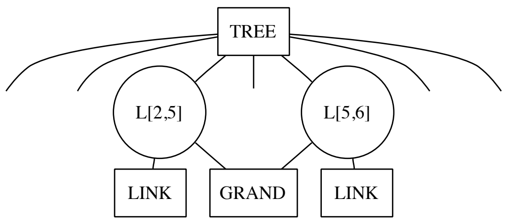


Figure 1: A fragment of a factor graph, illustrating a few of the unary, binary, and global factors that affect variables L_{25} and L_{56}. The GRAND factor induces a loop.

eralization of the forward-backward algorithm that is deeply studied in the graphical models literature (Yedidia et al., 2004, for example). We briefly sketch the method in terms of our parsing task.

4.1 Where BP comes from

The basic BP idea is simple. Variable L_{34} maintains a *distribution* over values true and false—a "belief"—that is periodically recalculated based on the current *distributions* at other variables.[8]

Readers familiar with Gibbs sampling can regard this as a kind of deterministic approximation. In Gibbs sampling, L_{34}'s *value* is periodically resampled based on the current *values* of other variables. Loopy BP works not with random samples but their expectations. Hence it is approximate but tends to converge much faster than Gibbs sampling will mix.

It is convenient to visualize an undirected **factor graph** (Fig. 1), in which each factor is connected to the variables it depends on. Many factors may connect to—and hence influence—a given variable such as L_{34}. If X is a variable *or* a factor, $\mathcal{N}(X)$ denotes its set of neighbors.

4.2 What BP accomplishes

Given an input sentence W and a parameter vector θ, the collection of factors F_m defines a probability distribution (1). The parser should determine the values of the individual variables. In other words, we would like to marginalize equation (1) to obtain the distribution $p(L_{34})$ over L_{34} = true vs. false, the distribution $p(T_4)$ over tags, etc.

If the factor graph is *acyclic*, then BP computes these marginal distributions exactly. Given an HMM, for example, BP reduces to the forward-backward algorithm.

BP's estimates of these distributions are called **beliefs about the variables**. BP also computes **beliefs about the factors**, which are useful in learning θ (see §7). E.g., if the model includes the factor TAGLINK$_{ij}$, which is connected to variables L_{ij}, T_i, T_j, then BP will estimate the marginal joint distribution $p(L_{ij}, T_i, T_j)$ over (boolean, tag, tag) triples.

When the factor graph has loops, BP's beliefs are usually not the true marginals of equation (1) (which are in general intractable to compute). Indeed, BP's beliefs may not be the true marginals of *any* distribution $p(\mathcal{A})$ over assignments, i.e., they may be globally inconsistent. All BP does is to incrementally adjust the beliefs till they are at least *locally* consistent: e.g., the beliefs at factors TAGLINK$_{ij}$ and TAGLINK$_{ik}$ must both imply[9] the same belief about variable T_i, their common neighbor.

4.3 The BP algorithm

This iterated negotiation among the factors is handled by **message passing** along the edges of the factor graph. A **message** to or from a variable is a (possibly unnormalized) probability distribution over the values of that variable.

The variable V sends a message to factor F, saying "My *other* neighboring factors G jointly suggest that I have posterior distribution $q_{V\to F}$ (assuming that they are sending me independent evidence)." Meanwhile, factor F sends messages to V, saying, "Based on my factor function and the messages received from my *other* neighboring variables U about *their* values (and assuming that those messages are independent), I suggest you have posterior distribution $r_{F\to V}$ over *your* values."

To be more precise, BP at each iteration k (until convergence) updates two kinds of messages:

$$q_{V\to F}^{(k+1)}(v) = \kappa \prod_{G\in\mathcal{N}(V)\setminus F} r_{G\to V}^{(k)}(v) \tag{3}$$

from variables to factors, and

$$r_{F\to V}^{(k+1)}(v) = \kappa \sum_{\mathcal{A} \text{ s.t. } \mathcal{A}[V]=v} F(\mathcal{A}) \prod_{U\in\mathcal{N}(F)\setminus V} q_{U\to F}^{(k)}(\mathcal{A}[U]) \tag{4}$$

[8]Or, more precisely—this is the tricky part—based on versions of those other distributions that do not factor in L_{34}'s reciprocal influence on *them*. This prevents (e.g.) L_{34} and T_3 from mutually reinforcing each other's existing beliefs.

[9]In the sense that marginalizing the belief $p(L_{ij}, T_i, T_j)$ at the factor yields the belief $p(T_i)$ at the variable.

from factors to variables. Each message is a probability distribution over values v of V, normalized by a scaling constant κ. Alternatively, messages may be left as unnormalized distributions, choosing $\kappa \neq 1$ only as needed to prevent over- or underflow. Messages are initialized to uniform distributions.

Whenever we wish, we may compute the beliefs at V and F:

$$b^{(k+1)}_{V\rightarrow}(v) \overset{\text{def}}{=} \kappa \prod_{G \in \mathcal{N}(V)} r^{(k)}_{G\rightarrow V}(v) \tag{5}$$

$$b^{(k+1)}_{F\rightarrow}(\mathcal{A}) \overset{\text{def}}{=} \kappa\, F(\mathcal{A}) \prod_{U \in \mathcal{N}(F)} q^{(k)}_{U\rightarrow F}(\mathcal{A}[U]) \tag{6}$$

These beliefs do not truly characterize the expected behavior of Gibbs sampling (§4.1), since the products in (5)–(6) make conditional independence assumptions that are valid only if the factor graph is acyclic. Furthermore, on cyclic ("loopy") graphs, BP might only converge to a local optimum (Weiss and Freedman, 2001), or it might not converge at all. Still, BP often leads to good, fast approximations.

5 Achieving Low Asymptotic Runtime

One iteration of standard BP simply updates all the messages as in equations (3)–(4): one message per edge of the factor graph.

Therefore, adding new factors to the model increases the runtime per iteration *additively*, by increasing the number of messages to update. We believe this is a compelling advantage over dynamic programming—in which new factors usually increase the runtime and space *multiplicatively* by exploding the number of distinct items.[10]

5.1 Propagators for local constraints

But how long does updating each message take? The runtime of summing over all assignments $\sum_{\mathcal{A}}$ in equation (4) may appear prohibitive. Crucially, however, $F(\mathcal{A})$ only depends on the values in $\mathcal{A}$ of F's its neighboring variables $\mathcal{N}(F)$. So this sum is proportional to a sum over *restricted* assignments to just those variables.[11]

For example, computing a message from TAGLINK$_{ij}$ $\rightarrow T_i$ only requires iterating over all (boolean, tag, tag) triples.[12] The runtime to update that message is therefore $O(2 \cdot |\mathcal{T}| \cdot |\mathcal{T}|)$.

5.2 Propagators for global constraints

The above may be tolerable for a ternary factor. But how about global factors? EXACTLY1$_j$ has n neighboring boolean variables: surely we cannot iterate over all 2^n assignments to these! TREE is even worse, with $2^{O(n^2)}$ assignments to consider. We will give specialized algorithms for handling these summations more efficiently.

A historical note is in order. Traditional constraint satisfaction corresponds to the special case of (1) where *all* factors F_m are hard constraints (with values in $\{0, 1\}$). In that case, loopy BP reduces to an algorithm for generalized arc consistency (Mackworth, 1977; Bessière and Régin, 1997; Dechter, 2003), and updating a factor's outgoing messages is known as **constraint propagation**. Régin (1994) famously introduced an efficient propagator for a global constraint, ALLDIFFERENT, by adapting combinatorial bipartite matching algorithms.

In the same spirit, we will demonstrate efficient propagators for our global constraints, e.g. by adapting combinatorial algorithms for *weighted parsing*. We are unaware of any previous work on global factors in sum-product BP, although for *max-product* BP,[13] Duchi et al. (2007) independently showed that a global 1-to-1 alignment constraint—a kind of weighted ALLDIFFERENT—permits an efficient propagator based on weighted bipartite matching.

5.3 Constraint propagators for parsing

Table 1 shows our asymptotic runtimes for all factors in §§3.3–3.4. Remember that if several of these

[10]For example, with unknown tags T, a model with PTREE+TAGLINK will take only $O(n^3 + n^2g^2)$ time for BP, compared to $O(n^3g^2)$ time for dynamic programming (Eisner & Satta 1999). Adding TRIGRAM, which is string-local rather than tree-local, will increase this only to $O(n^3 + n^2g^2 + ng^3)$, compared to $O(n^3g^6)$ for dynamic programming.

Even more dramatic, adding the SIB family of $O(n^3)$ PAIR$_{ij,ik}$ factors will add only $O(n^3)$ to the runtime of BP (Table 1). By contrast, the runtime of dynamic programming becomes exponential, because each item must record its headword's full *set* of current children.

[11]The constant of proportionality may be folded into κ; it is the number of assignments to the *other* variables.

[12]Separately for *each* value v of T_i, get v's probability by summing over assignments to (L_{ij}, T_i, T_j) s.t. $T_i = v$.

[13]Max-product replaces the sums in equations (3)–(6) with maximizations. This replaces the forward-backward algorithm with its Viterbi approximation.

factor family	degree (each)	runtime (each)	count	runtime (total)
TREE	$O(n^2)$	$O(n^3)$	1	$O(n^3)$
PTREE	$O(n^2)$	$O(n^3)$	1	$O(n^3)$
EXACTLY1	$O(n)$	$O(n)$	n	$O(n^2)$
ATMOST1	$O(n)$	$O(n)$	n	$O(n^2)$
NOT2	2	$O(1)$	$O(n^3)$	$O(n^3)$
NO2CYCLE	2	$O(1)$	$O(n^2)$	$O(n^2)$
LINK	1	$O(1)$	$O(n^2)$	$O(n^2)$
GRAND	2	$O(1)$	$O(n^3)$	$O(n^3)$
SIB	2	$O(1)$	$O(n^3)$	$O(n^3)$
CHILDSEQ	$O(n)$	$O(n^2)$	$O(n)$	$O(n^3)$
NOCROSS	$O(n)$	$O(n)$	$O(n^2)$	$O(n^3)$
TAG	1	$O(g)$	$O(n)$	$O(ng)$
TAGLINK	3	$O(g^2)$	$O(n^2)$	$O(n^2g^2)$
TRIGRAM	$O(n)$	$O(ng^3)$	1	$O(ng^3)$

Table 1: Asymptotic runtimes of the propagators for various factors (where n is the sentence length and g is the size of the tag set $\mathcal{T}$). An iteration of standard BP propagates through each factor once. Running a factor's propagator will update all of its outgoing messages, based on its current incoming messages.

factors are included, the total runtime is *additive*.[14]

Propagating the local factors is straightforward (§5.1). We now explain how to handle the global factors. *Our main trick is to work backwards from marginal beliefs.* Let F be a factor and V be one of its neighboring variables. At any time, F has a marginal belief about V (see footnote 9),

$$b^{(k+1)}_{F\rightarrow}(V = v) = \sum_{\mathcal{A} \text{ s.t. } \mathcal{A}[V]=v} b^{(k+1)}_{F\rightarrow}(\mathcal{A}) \qquad (7)$$

a sum over (6)'s products of incoming messages. By the definition of $r_{F\rightarrow V}$ in (4), and distributivity, we can also express the marginal belief (7) as a pointwise product of outgoing and incoming messages[15]

$$b^{(k+1)}_{F\rightarrow}(V = v) = r^{(k+1)}_{F\rightarrow V}(v) \cdot q^{(k)}_{V\rightarrow F}(v) \qquad (8)$$

up to a constant. If we can quickly sum up the marginal belief (7), then (8) says we can divide out *each* particular incoming message $q^{(k)}_{V\rightarrow F}$ to obtain its corresponding outgoing message $r^{(k+1)}_{F\rightarrow V}$.

[14]We may ignore the cost of propagators at the variables. Each outgoing message from a variable can be computed in time proportional to its size, which may be amortized against the cost of generating the corresponding incoming message.

[15]E.g., the familiar product of forward and backward messages that is used to extract posterior marginals from an HMM.

Note that the marginal belief and both messages are unnormalized distributions over values v of V. F and k are clear from context below, so we simplify the notation so that (7)–(8) become

$$b(V = v) \;=\; \sum_{\mathcal{A} \text{ s.t. } \mathcal{A}[V]=v} b(\mathcal{A}) \;=\; r_V(v) \cdot q_V(v)$$

TRIGRAM must sum over assignments to the tag sequence T. The belief (6) in a given assignment is a product of trigram scores (which play the role of transition weights) and incoming messages q_{T_j} (playing the role of emission weights). The marginal belief (7) needed above, $b(T_i = t)$, is found by summing over assignments where $T_i = t$. All marginal beliefs are computed together in $O(ng^3)$ total time by the forward-backward algorithm.[16]

EXACTLY1$_j$ is a *sparse* hard constraint. Even though there are 2^n assignments to its n neighboring variables $\{L_{ij}\}$, the factor function returns 1 on only n assignments and 0 on the rest. In fact, for a given i, $b(L_{ij} = \mathsf{true})$ in (7) is defined by (6) to have exactly one non-zero summand, in which $\mathcal{A}$ puts $L_{ij} = \mathsf{true}$ and all other $L_{i'j} = \mathsf{false}$. We compute the marginal beliefs for all i together in $O(n)$ total time:

1. Pre-compute $\pi \overset{\text{def}}{=} \prod_i q_{L_{ij}}(\mathsf{false})$.[17]
2. For each i, compute the marginal belief $b(L_{ij} = \mathsf{true})$ as $\pi \cdot \bar{q}_{L_{ij}}$, where $\bar{q}_{L_{ij}} \in \mathbb{R}$ denotes the odds ratio $q_{L_{ij}}(\mathsf{true})/q_{L_{ij}}(\mathsf{false})$.[18]
3. The partition function $b()$ denotes $\sum_{\mathcal{A}} b(\mathcal{A})$; compute it in this case as $\sum_i b(L_{ij} = \mathsf{true})$.
4. For each i, compute $b(L_{ij} = \mathsf{false})$ by subtraction, as $b() - b(L_{ij} = \mathsf{true})$.

TREE and PTREE must sum over assignments to the $O(n^2)$ neighboring variables $\{L_{ij}\}$. There are now exponentially many non-zero summands, those in which $\mathcal{A}$ corresponds to a valid tree. Nonetheless,

[16]Which is itself an exact BP algorithm, but on a different graph—a junction tree formed from the graph of TRIGRAM subfactors. Each variable in the junction tree is a *bigram*. If we had simply replaced the global TRIGRAM factor with its subfactors in the full factor graph, we would have had to resort to Generalized BP (Yedidia et al., 2004) to obtain the same exact results.

[17]But taking $\pi = 1$ gives the same results, up to a constant.

[18]As a matter of implementation, this odds ratio $\bar{q}_{L_{ij}}$ can be used to represent the incoming message $q_{L_{ij}}$ everywhere.

we can follow the same approach as for EXACTLY1. Steps 1 and 4 are modified to iterate over all i, j such that L_{ij} is a variable. In step 3, the partition function $\sum_{\mathcal{A}} b(\mathcal{A})$ is now π times the total weight of all trees, where the weight of a given tree is the product of the $\bar{q}_{L_{ij}}$ values of its n edges. In step 2, the marginal belief $b(L_{ij} = \text{true})$ is now π times the total weight of all trees having edge $i \rightarrow j$.

We perform these combinatorial sums *by calling a first-order parsing algorithm,* with edge weights $\bar{q}_{ij}$. Thus, as outlined in §2, a first-order parser is called each time we propagate through the global TREE or PTREE constraint, using edge weights that include the first-order LINK factors but also multiply in any current messages from higher-order factors.

The parsing algorithm simultaneously computes the partition function $b()$, and all $O(n^2)$ marginal beliefs $b(L_{ij} = \text{true})$. For PTREE (projective), it is the inside-outside version of a dynamic programming algorithm (Eisner, 1996). For TREE (non-projective), Koo et al. (2007) and Smith and Smith (2007) show how to employ the matrix-tree theorem. In both cases, the total time is $O(n^3)$.[19]

NOCROSS$_{j\ell}$ must sum over assignments to $O(n)$ neighboring variables $\{L_{ij}\}$ and $\{L_{k\ell}\}$. The non-zero summands are assignments where j and ℓ each have exactly one parent. At step 1, $\pi \overset{\text{def}}{=} \prod_i q_{L_{ij}}(\text{false}) \cdot \prod_k q_{L_{k\ell}}(\text{false})$. At step 2, the marginal belief $b(L_{ij} = \text{true})$ sums over the n non-zero assignments containing $i \rightarrow j$. It is $\pi \cdot \bar{q}_{L_{ij}} \cdot \sum_k \bar{q}_{L_{k\ell}} \cdot \text{PAIR}_{ij,k\ell}$, where $\text{PAIR}_{ij,k\ell}$ is $x_{j\ell}$ if $i \rightarrow j$ crosses $k \rightarrow \ell$ and is 1 otherwise. $x_{j\ell}$ is some factor value defined by equation (2) to penalize or reward the crossing. Steps 3–4 are just as in EXACTLY1$_j$.

The question is how to compute $b(L_{ij} = \text{true})$ for each i in only $O(1)$ time,[20] so that we can propagate each of the $O(n^2)$ NOCROSS$_{j\ell}$ in $O(n)$ time. This is why we allowed $x_{j\ell}$ to depend only on j, ℓ. We can rewrite the sum $b(L_{ij} = \text{true})$ as

$$\pi \cdot \bar{q}_{L_{ij}} \cdot (x_{j\ell} \cdot \sum_{\text{crossing } k} \bar{q}_{L_{k\ell}} + 1 \cdot \sum_{\text{noncrossing } k} \bar{q}_{L_{k\ell}}) \quad (9)$$

To find this in $O(1)$ time, we precompute for each ℓ an array of partial sums $Q_\ell[s,t] \overset{\text{def}}{=} \sum_{s \le k \le t} \bar{q}_{L_{k\ell}}$. Since $Q_\ell[s,t] = Q_\ell[s,t-1] + \bar{q}_{L_{t\ell}}$, we can compute each entry in $O(1)$ time. The total precomputation time over all ℓ, s, t is then $O(n^3)$, with the array Q_ℓ shared across all factors NOCROSS$_{j'\ell}$. The crossing sum is respectively $Q_\ell[0, i-1] + Q_\ell[j+1, n]$, $Q_\ell[i+1, j-1]$, or 0 according to whether $\ell \in (i,j)$, $\ell \notin [i,j]$, or $\ell = i$.[21] The non-crossing sum is $Q_\ell[0,n]$ minus the crossing sum.

CHILDSEQ$_i$, like TRIGRAM, is propagated by a forward-backward algorithm. In this case, the algorithm is easiest to describe by replacing CHILDSEQ$_i$ in the factor graph by a collection of local subfactors, which pass messages in the ordinary way.[22] Roughly speaking,[23] at each $j \in [1,n]$, we introduce a new variable C_{ij}—a hidden state whose value is the position of i's previous child, if any (so $0 \le C_{ij} < j$). So the ternary subfactor on $(C_{ij}, L_{ij}, C_{i,j+1})$ has value 1 if $L_{ij} = \text{false}$ and $C_{i,j+1} = C_{i,j}$; a sibling-bigram score ($\text{PAIR}_{iC_{ij}, iC_{i,j+1}}$) if $L_{ij} = \text{true}$ and $C_{i,j+1} = j$; and 0 otherwise. The sparsity of this factor, which is 0 almost everywhere, is what gives CHILDSEQ$_i$ a total runtime of $O(n^2)$ rather than $O(n^3)$. It is equivalent to forward-backward on an HMM with n observations (the L_{ij}) and n states per observation (the C_j), with a *deterministic* (thus sparse) transition function.

6 Decoding Trees

BP computes local beliefs, e.g. the conditional probability that a link L_{ij} is present. But if we wish to output a single well-formed dependency tree, we need to find a single assignment to all the $\{L_{ij}\}$ that satisfies the TREE (or PTREE) constraint.

Our final belief about the TREE *factor* is a distribution over such assignments, in which a tree's probability is proportional to the probability of its edge weights $\bar{q}_{L_{ij}}$ (incoming messages). We could simply return the mode of this distribution (found by using a 1-best first-order parser) or the k-best trees, or take samples.

[19]A dynamic algorithm could incrementally update the outgoing messages if only a few incoming messages have changed (as in asynchronous BP). In the case of TREE, dynamic matrix inverse allows us to update any row or column (i.e., messages from all parents or children of a given word) and find the new inverse in $O(n^2)$ time (Sherman and Morrison, 1950).

[20]Symmetrically, we compute $b(L_{k\ell} = \text{true})$ for each k.

[21]There are no NOCROSS$_{j\ell}$ factors with $\ell = j$.

[22]We still treat CHILDSEQ$_i$ as a global factor and compute all its correct outgoing messages on a single BP iteration, via *serial* forward and backward sweeps through the subfactors. Handling the subfactors in parallel, (3)–(4), would need $O(n)$ iterations.

[23]Ignoring the treatment of boundary symbols "#" (see §3.4).

In our experiments, we actually take the edge weights to be not the messages $\bar{q}_{L_{ij}}$ from the links, but the full beliefs $\bar{b}_{L_{ij}}$ at the links (where $\bar{b}_{L_{ij}} \overset{\text{def}}{=} \log b_{L_{ij}}(\textsf{true})/b_{L_{ij}}(\textsf{false})$). These are passed into a fast algorithm for maximum spanning tree (Tarjan, 1977) or maximum projective spanning tree (Eisner, 1996). This procedure is equivalent to minimum Bayes risk (MBR) parsing (Goodman, 1996) with a dependency accuracy loss function.

Notice that the above decoding approaches do not enforce any hard constraints other than TREE in the final output. In addition, they only recover values of the L_{ij} variables. They marginalize over other variables such as tags and link roles. This solves the problem of "nuisance" variables (which merely fragment probability mass among refinements of a parse). On the other hand, it may be undesirable for variables whose values we desire to recover.[24]

7 Training

Our training method also uses beliefs computed by BP, but at the *factors*. We choose the weight vector θ by maximizing the log-probability of training data under equation (1), regularizing only by early stopping. If all variables are observed in training, this objective function is *convex* (as for any log-linear model).

The difficult step in computing the gradient of our objective is finding $\nabla_\theta \log Z$, where Z in equation (1) is the normalizing constant (partition function) that sums over all assignments $\mathcal{A}$. (Recall that Z, like each F_m, depends implicitly on W and θ.) As usual for log-linear models,

$$\nabla_\theta \log Z = \sum_m \mathbf{E}_{p(\mathcal{A})}[\nabla_\theta F_m(\mathcal{A})] \qquad (10)$$

Since $\nabla_\theta F_m(\mathcal{A})$ only depends on the assignment $\mathcal{A}$'s values for variables that are connected to F_m in the factor graph, its expectation under $p(\mathcal{A})$ depends only on the marginalization of $p(\mathcal{A})$ to those variables jointly. Fortunately, BP provides an estimate of that marginal distribution, namely, its belief about the factor F_m, given W and θ (§4.2).[25]

Note that the hard constraints do not depend on θ at all; so their summands in equation (10) will be 0.

We employ stochastic gradient descent (Bottou, 2003), since this does not require us to compute the objective function itself but only to (approximately) estimate its gradient as explained above. Alternatively, given any of the MAP decoding procedures from §6, we could use an error-driven learning method such as the perceptron or MIRA.[26]

8 Experiments

We asked: (1) For projective parsing, where higher-order factors have traditionally been incorporated into slow but exact dynamic programming (DP), what are the comparative speed and quality of the BP approximation? (2) How helpful are such higher-order factors—particularly for non-projective parsing, where BP is needed to make them tractable? (3) Do our global constraints (e.g., TREE) contribute to the goodness of BP's approximation?

[24]An alternative is to attempt to find the most probable ("MAP") assignment to all variables—using the max-product algorithm (footnote 13) or one of its recent variants. The estimated marginal beliefs become "max marginals," which assess the 1-best assignment consistent with each value of the variable.

We can indeed build max-product propagators for our global constraints. PTREE still propagates in $O(n^3)$ time: simply change the first-order parser's semiring (Goodman, 1999) to use max instead of sum. TREE requires $O(n^4)$ time: it seems that the $O(n^2)$ max marginals must be computed separately, each requiring a separate call to an $O(n^2)$ maximum spanning tree algorithm (Tarjan, 1977).

If max-product BP converges, we may simply output each variable's favorite value (according to its belief), if unique. However, max-product BP tends to be unstable on loopy graphs, and we may not wish to wait for full convergence in any case. A more robust technique for extracting an assignment is to mimic Viterbi decoding, and "follow backpointers" of the max-product computation along some spanning subtree of the factor graph.

A slower but potentially more stable alternative is deterministic annealing. Replace each factor $F_m(\mathcal{A})$ with $F_m(\mathcal{A})^{1/T}$, where $T > 0$ is a **temperature**. As $T \to 0$ ("quenches"), the distribution (1) retains the same mode (the MAP assignment), but becomes more sharply peaked at the mode, and sum-product BP approaches max-product BP. Deterministic annealing runs sum-product BP while gradually reducing T toward 0 as it iterates. By starting at a high T and reducing T slowly, it often manages in practice to find a good local optimum. We may then extract an assignment just as we do for max-product.

[25]One could use coarser estimates at earlier stages of training, by running fewer iterations of BP.

[26]The BP framework makes it tempting to extend an MRF model with various sorts of latent variables, whose values are not specified in training data. It is straightforward to train under these conditions. When counting which features fire on a training parse or (for error-driven training) on an current erroneous parse, we can find *expected* counts if these parses are not fully observed, by using BP to sum over latent variables.

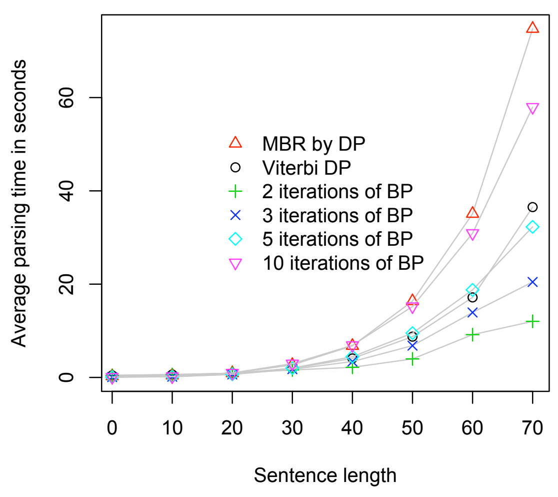


Figure 2: Runtime of BP parser on various sentence lengths compared to $O(n^4)$ dynamic programming.

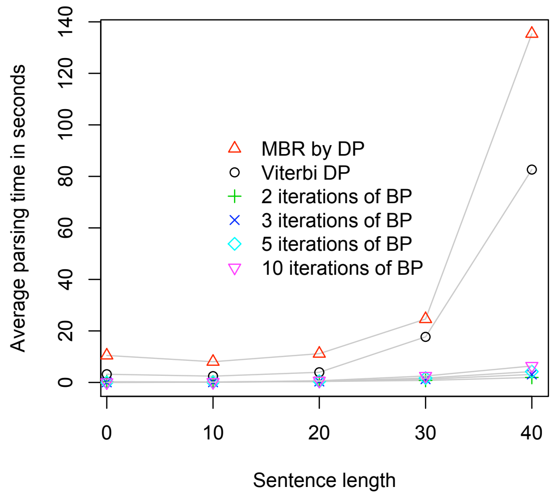


Figure 3: Runtime of BP parser on various sentence lengths compared to $O(n^5)$ dynamic programming. DP is so slow for length > 45 that we do not even show it.

8.1 Data

We trained and tested on three languages from the CoNLL Dependency Parsing Shared Task (Nivre et al., 2007). The English data for that task were converted from the Penn Treebank to dependencies using a trace-recovery algorithm that induced some very slight non-projectivity—about 1% of links crossed other links. Danish is a slightly more non-projective language (3% crossing links). Dutch is the most non-projective language in the corpus (11%). In all cases, the test input W consists of part-of-speech-tagged words, so T variables were not used.

8.2 Features

Although BP makes it cheap to incorporate many non-local features and latent variables at once, we kept our models relatively simple in this paper.

Our first-order LINK$_{ij}$ factors replicate McDonald et al. (2005). Following equation (2), they are defined using binary features that look at words i and j, the distance $j - i$, and the tags (provided in W) of words at, around, and between i and j.

Our second-order features are similar. In the GRAND factors, features fire for particular triples of tags and of coarse tags. A feature also fires if the grandparent falls between the child and parent, inducing crossing dependency links. The CHILD-SEQ factors included features for tags, and likewise coarse tags, on adjacent sibling pairs and parent-sibling-sibling triples. Each of these features also have versions that were conjoined with link direction—pairs of directions in the grandparent case—or with signed link length of the child or farther sibling. Lengths were binned per McDonald et al. (2005). The NOCROSS$_{j\ell}$ factors consider the tag and coarse tag attributes of the two child words j and ℓ, separately or jointly.

8.3 Experimental procedures

We trained all models using stochastic gradient descent (§7). SGD initialized $\vec{\theta} = 0$ and ran for 10 consecutive passes over the data; we picked the stopping point that performed best on held-out data.

When comparing runtimes for projective parsers, we took care to produce comparable implementations. All beliefs and dynamic programming items were stored and indexed using the high-level Dyna language,[27] while all inference and propagation was written in C++. The BP parser averaged 1.8 seconds per sentence for non-projective parsing and 1.5 seconds per sentence for projective parsing (1.2 and 0.9 seconds/sentence for $\leq$ 40 words), using our standard setup, which included five iterations of BP and the final MBR tree decoding pass.

In our tables, we boldface the best result in each column along with any results that are not significantly worse (paired permutation test, $p < .05$).

[27]This dominates runtime, and probably slows down all our parsers by a factor of 4–11 owing to known inefficiencies in the Dyna prototype we used (Eisner et al., 2005).

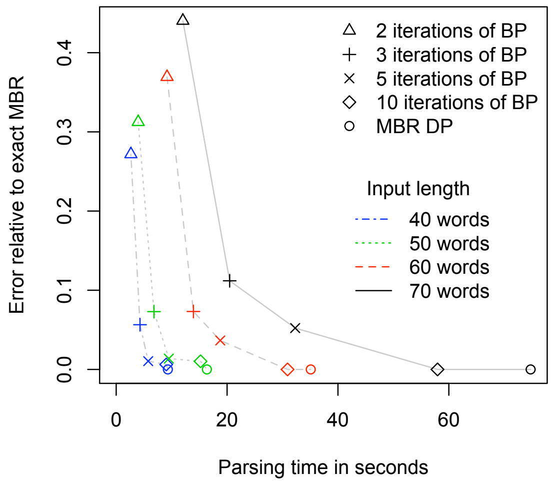


Figure 4: Runtime vs. search error after different numbers of BP iterations. This shows the simpler model of Fig. 2, where DP is still relatively fast.

8.4 Faster higher-order projective parsing

We built a first-order projective parser—one that uses only factors PTREE and LINK—and then compared the cost of incorporating second-order factors, GRAND and CHILDSEQ, by BP versus DP.[28]

Under DP, the first-order runtime of $O(n^3)$ is increased to $O(n^4)$ with GRAND, and to $O(n^5)$ when we add CHILDSEQ as well. BP keeps runtime down to $O(n^3)$—although with a higher constant factor, since it takes several rounds to converge, and since it computes more than just the best parse.[29]

Figures 2–3 compare the empirical runtimes for various input sentence lengths. With only the GRAND factor, exact DP can still find the Viterbi parse (though not the MBR parse[29]) faster than ten iterations of the asymptotically better BP (Fig. 2), at least for sentences with $n \leq 75$. However, once we add the CHILDSEQ factor, BP is always faster—dramatically so for longer sentences (Fig. 3). More complex models would widen BP's advantage.

Fig. 4 shows the tradeoff between runtime and search error of BP in the former case (GRAND only). To determine BP's search error at finding the MBR parse, we measured its dependency accuracy not against the gold standard, but against the optimal MBR parse under the model, which DP is able to find. After 10 iterations, the overall macro-averaged search error compared to $O(n^4)$ DP MBR is 0.4%; compared to $O(n^5)$ (not shown), 2.4%. More BP iterations may help accuracy. In future work, we plan to compare BP's speed-accuracy curve on more complex projective models with the speed-accuracy curve of *pruned* or *reranked* DP.

[28]We trained these parsers using exact DP, using the inside-outside algorithm to compute equation (10). The training and test data were English, and for this section we filtered out sentences with non-projective links.

[29]Viterbi parsing in the log domain only needs the $(\max, +)$ semiring, whereas both BP and any MBR parsing must use the slower $(+, \log+)$ so that they can compute marginals.

		Danish	Dutch	English
(a)	TREE+LINK	85.5	87.3	88.6
	+NOCROSS	86.1	88.3	89.1
	+GRAND	86.1	**88.6**	89.4
	+CHILDSEQ	**86.5**	**88.5**	90.1
(b)	Proj. DP	86.0	84.5	**90.2**
	+hill-climbing	86.1	87.6	**90.2**

Table 2: (a) Percent unlabeled dependency accuracy for various non-projective BP parsers (5 iterations only), showing the cumulative contribution of different features. (b) Accuracy for an projective DP parser with all features. For relatively non-projective languages (Danish and especially Dutch), the exact projective parses can be improved by non-projective hill-climbing—but in those cases, just running our non-projective BP is better and faster.

8.5 Higher-order non-projective parsing

The BP approximation can be used to improve the accuracy of *non*-projective parsing by adding higher-order features. These would be NP-hard to incorporate exactly; DP cannot be used.

We used BP with a non-projective TREE factor to train conditional log-linear parsing models of two highly non-projective languages, Danish and Dutch, as well as slightly non-projective English (§8.1). In all three languages, the first-order non-projective parser greatly overpredicts the number of crossing links. We thus added NOCROSS factors, as well as GRAND and CHILDSEQ as before. All of these significantly improve the first-order baseline, though not necessarily cumulatively (Table 2).

Finally, Table 2 compares loopy BP to a previously proposed "hill-climbing" method for approximate inference in non-projective parsing McDonald and Pereira (2006). Hill-climbing decodes our richest non-projective model by finding the best *projective* parse under that model—using slow, *higher-order* DP—and then greedily modifies words' parents until the parse score (1) stops improving.

Decoding	Danish	Dutch	English
NOT2	81.8 (76.7)	83.3 (75.0)	87.5 (66.4)
ATMOST1	85.4 (82.2)	**87.3** (86.3)	88.5 (84.6)
EXACTLY1	**85.7** (85.0)	87.0 (86.7)	88.6 (86.0)
+ NO2CYCLE	85.0 (**85.2**)	86.2 (86.7)	88.5 (86.2)
TREE	**85.5** (**85.5**)	**87.3** (**87.3**)	88.6 (**88.6**)
PTREE	**85.8**	83.9	**88.8**

Table 3: After training a non-projective first-order model with TREE, decoding it with weaker constraints is asymptotically faster (except for NOT2) but usually harmful. (Parenthetical numbers show that the harm is compounded if the weaker constraints are used in training as well; even though this matches training to test conditions, it may suffer more from BP's approximate gradients.) Decoding the TREE model with the even stronger PTREE constraint can actually be helpful for a more projective language. All results use 5 iterations of BP.

BP for non-projective languages is much faster and more accurate than the hill-climbing method. Also, hill-climbing only produces an (approximate) 1-best parse, but BP also obtains (approximate) marginals of the distribution over *all* parses.

8.6 Importance of global hard constraints

Given the BP architecture, do we even need the hard TREE constraint? Or would it suffice for more local hard constraints to negotiate locally via BP?

We investigated this for non-projective first-order parsing. Table 3 shows that global constraints are indeed important, and that it is essential to use TREE during training. At test time, the weaker but still global EXACTLY1 may suffice (followed by MBR decoding to eliminate cycles), for total time $O(n^2)$.

Table 3 includes NOT2, which takes $O(n^3)$ time, merely to demonstrate how the BP approximation becomes more accurate for training and decoding when we join the simple NOT2 constraints into more global ATMOST1 constraints. This does not change the distribution (1), but makes BP enforce stronger local consistency requirements at the factors, relying less on independence assumptions. In general, one can get better BP approximations by replacing a group of factors $F_m(\mathcal{A})$ with their product.[30]

The above experiments concern *gold-standard* accuracy under a given *first-order*, *non-projective* model. Flipping all three of these parameters for Danish, we confirmed the pattern by instead measuring *search error* under a *higher-order*, *projective* model (PTREE+LINK+GRAND), when PTREE was weakened during decoding. Compared to the MBR parse under that model, the search errors from decoding with weaker hard constraints were 2.2% for NOT2, 2.1% for EXACTLY1, 1.7% for EXACTLY1 + NO2CYCLE, and 0.0% for PTREE.

9 Conclusions and Future Work

Belief propagation improves *non-projective* dependency parsing with features that would make exact inference intractable. For *projective* parsing, it is significantly faster than exact dynamic programming, at the cost of small amounts of search error,

We are interested in extending these ideas to phrase-structure and lattice parsing, and in trying other higher-order features, such as those used in parse reranking (Charniak and Johnson, 2005; Huang, 2008) and history-based parsing (Nivre and McDonald, 2008). We could also introduce new variables, e.g., nonterminal refinements (Matsuzaki et al., 2005), or secondary links M_{ij} (not constrained by TREE/PTREE) that augment the parse with representations of control, binding, etc. (Sleator and Temperley, 1993; Buch-Kromann, 2006).

Other parsing-like problems that could be attacked with BP appear in syntax-based machine translation. Decoding is very expensive with a synchronous grammar composed with an n-gram language model (Chiang, 2007)—but our footnote 10 suggests that BP might incorporate a language model rapidly. String alignment with synchronous grammars is quite expensive even for simple synchronous formalisms like ITG (Wu, 1997)—but Duchi et al. (2007) show how to incorporate bipartite matching into max-product BP.

Finally, we can take advantage of improvements to BP proposed in the context of other applications. For example, instead of updating all messages in parallel at every iteration, it is empirically faster to serialize updates using a priority queue (Elidan et al., 2006; Sutton and McCallum, 2007).[31]

[30]In the limit, one could replace the product (1) with a single all-purpose factor; then BP would be exact—but slow. (In constraint satisfaction, joining constraints similarly makes arc consistency slower but better at eliminating impossible values.)

[31]These methods need alteration to handle our global propagators, which do update all their outgoing messages at once.

References

C. Bessière and J.-C. Régin. 1997. Arc consistency for general constraint networks: preliminary results. In *IJCAI*, pages 398–404.

L. Bottou. 2003. Stochastic learning. In *Advanced Lectures in Machine Learning*, pages 146–168. Springer.

A. Braunstein, M. Mezard, and R. Zecchina. 2005. Survey propagation: An algorithm for satisfiability. *Random Structures and Algorithms*, 27:201–226.

M. Buch-Kromann. 2006. *Discontinuous Grammar. A Model of Human Parsing and Language Acquisition"*. Dr.ling.merc. dissertation, Copenhagen Business School.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *ACL*, pages 173–180.

D. Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

R. Dechter. 2003. *Constraint Processing*. Morgan Kaufmann.

J. Duchi, D. Tarlow, G. Elidan, and D. Koller. 2007. Using combinatorial optimization within max-product belief propagation. In *NIPS 2006*, pages 369–376.

J. Eisner and G. Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *ACL*, pages 457–480.

J. Eisner, E. Goldlust, and N. A. Smith. 2005. Compiling comp ling: Weighted dynamic programming and the dyna language. In *HLT-EMNLP*, pages 281–290.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *COLING*.

G. Elidan, I. McGraw, and D. Koller. 2006. Residual belief propagation: Informed scheduling for asynchronous message passing. In *UAI*.

J. T. Goodman. 1996. Parsing algorithms and metrics. In *ACL*, pages 177–183.

J. Goodman. 1999. Semiring parsing. *Computational Linguistics*, 25(4):573–605.

L. Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *ACL*, pages 586–594.

T. Koo, A. Globerson, X. Carreras, and M. Collins. 2007. Structured prediction models via the Matrix-Tree Theorem. In *EMNLP-CoNLL*.

D. MacKay. 2003. *Information Theory, Inference, and Learning Algorithms*. Cambridge.

A. Mackworth. 1977. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118.

A. F. T. Martins, D. Das, N. A. Smith, and E. P. Xing. 2008. Stacking dependency parsers. In *EMNLP*.

T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic CFG with latent annotations. In *ACL*, pages 75–82.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL*.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *ACL*.

J. Nivre and R. McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *ACL*.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*.

J.-C. Régin. 1994. A filtering algorithm for constraints of difference in csps. In *AAAI*, pages 362–367.

S. Riedel and J. Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *EMNLP*, pages 129–137.

J. Sherman and W. J. Morrison. 1950. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *Ann. Math. Stat.*, 21:124–127.

D. Sleator and D. Temperley. 1993. Parsing English with a link grammar. In *IWPT*, pages 277–291, August.

D. A. Smith and N. A. Smith. 2007. Probabilistic models of nonprojective dependency trees. In *EMNLP-CoNLL*.

C. Sutton and A. McCallum. 2004. Collective segmentation and labeling of distant entities in information extraction. In *ICML Workshop on Statistical Relational Learning*.

C. Sutton and A. McCallum. 2007. Improved dynamic schedules for belief propagation. In *UAI*.

C. Sutton, K. Rohanimanesh, and A. McCallum. 2004. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *ICML*.

R. E. Tarjan. 1977. Finding optimum branchings. *Networks*, 7:25–35.

R. W. Tromble and J. Eisner. 2006. A fast finite-state relaxation method for enforcing global constraints on sequence decoding. In *HLT-NAACL*, pages 423–430.

Y. Weiss and W. T. Freedman. 2001. On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47.

D. Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *CL*, 23(3):377–404.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. 2004. Constructing free-energy approximations and generalized belief approximation algorithms. MERL TR2004-040, Mitsubishi Electric Research Laboratories.

Stacking Dependency Parsers

André F. T. Martins[*†] **Dipanjan Das**[*] **Noah A. Smith**[*] **Eric P. Xing**[*]
[*]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA
[†]Instituto de Telecomunicações, Instituto Superior Técnico, Lisboa, Portugal
{afm,dipanjan,nasmith,epxing}@cs.cmu.edu

Abstract

We explore a stacked framework for learning to predict dependency structures for natural language sentences. A typical approach in graph-based dependency parsing has been to assume a factorized model, where local features are used but a global function is optimized (McDonald et al., 2005b). Recently Nivre and McDonald (2008) used the output of one dependency parser to provide features for another. We show that this is an example of *stacked learning*, in which a second predictor is trained to improve the performance of the first. Further, we argue that this technique is a novel way of approximating rich *non-local features* in the second parser, without sacrificing efficient, model-optimal prediction. Experiments on twelve languages show that stacking transition-based and graph-based parsers improves performance over existing state-of-the-art dependency parsers.

1 Introduction

In this paper we address a representation-efficiency tradeoff in statistical natural language processing through the use of **stacked learning** (Wolpert, 1992). This tradeoff is exemplified in **dependency parsing**, illustrated in Fig. 1, on which we focus in this paper:

- Exact algorithms for dependency parsing (Eisner and Satta, 1999; McDonald et al., 2005b) are tractable only when the model makes very strong, linguistically unsupportable independence assumptions, such as "arc factorization" for non-projective dependency parsing (McDonald and Satta, 2007).
- Feature-rich parsers must resort to search or greediness, (Ratnaparkhi et al., 1994; Sagae and Lavie, 2005; Hall et al., 2006), so that parsing solutions are inexact and learned models may be subject to certain kinds of bias (Lafferty et al., 2001).

A solution that leverages the complementary strengths of these two approaches—described in detail by McDonald and Nivre (2007)—was recently and successfully explored by Nivre and McDonald (2008). Our contribution begins by reinterpreting and generalizing their parser combination scheme as a **stacking of parsers**.

We give a new theoretical motivation for stacking parsers, in terms of extending a parsing model's feature space. Specifically, we view stacked learning as a way of approximating non-local features in a linear model, rather than making empirically dubious independence (McDonald et al., 2005b) or structural assumptions (e.g., projectivity, Eisner, 1996), using search approximations (Sagae and Lavie, 2005; Hall et al., 2006; McDonald and Pereira, 2006), solving a (generally NP-hard) integer linear program (Riedel and Clarke, 2006), or adding latent variables (Titov and Henderson, 2007). Notably, we introduce the use of very rich non-local *approximate* features in one parser, through the output of another parser. Related approaches are the belief propagation algorithm of Smith and Eisner (2008), and the "trading of structure for features" explored by Liang et al.

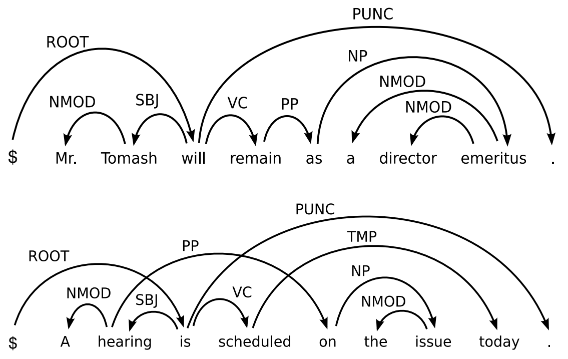

PUNC
ROOT
NP
NMOD
NMOD SBJ VC PP
NMOD
$ Mr. Tomash will remain as a director emeritus .
PUNC
TMP
ROOT PP
NP
NMOD SBJ VC
NMOD
$ A hearing is scheduled on the issue today .

Figure 1: A projective dependency parse (top), and a non-projective dependency parse (bottom) for two English sentences; examples from McDonald and Satta (2007).

(2008).

This paper focuses on dependency parsing, which has become widely used in relation extraction (Culotta and Sorensen, 2004), machine translation (Ding and Palmer, 2005), question answering (Wang et al., 2007), and many other NLP applications. We show that stacking methods outperform the approximate "second-order" parser of McDonald and Pereira (2006) on twelve languages and can be used within that approximation to achieve even better results. These results are similar in spirit to (Nivre and McDonald, 2008), but with the following novel contributions:

- a stacking interpretation,
- a richer feature set that includes non-local features (shown here to improve performance), and
- a variety of stacking architectures.

Using stacking with rich features, we obtain results competitive with Nivre and McDonald (2008) while preserving the fast quadratic parsing time of arc-factored spanning tree algorithms.

The paper is organized as follows. We discuss related prior work on dependency parsing and stacking in §2. Our model is given in §3. A novel analysis of stacking in linear models is given in §4. Experiments are presented in §5.

2 Background and Related Work

We briefly review work on the NLP task of **dependency parsing** and the machine learning framework known as **stacked learning**.

2.1 Dependency Parsing

Dependency syntax is a lightweight syntactic representation that models a sentence as a graph where the words are vertices and syntactic relationships are directed edges (arcs) connecting heads to their arguments and modifiers.

Dependency parsing is often viewed computationally as a structured prediction problem: for each input sentence x, with n words, exponentially many candidate dependency trees $y \in \mathcal{Y}(x)$ are possible in principle. We denote each tree by its set of vertices and directed arcs, $y = (V_y, A_y)$. A legal dependency tree has $n + 1$ vertices, each corresponding to one word plus a "wall" symbol, $, assumed to be the hidden root of the sentence. In a valid dependency tree, each vertex except the root has exactly one parent. In the *projective* case, arcs cannot cross when depicted on one side of the sentence; in the *non-projective* case, this constraint is not imposed (see Fig. 1).

2.1.1 Graph-based *vs.* transition-based models

Most recent work on dependency parsing can be categorized as *graph-based* or *transition-based*. In graph-based parsing, dependency trees are scored by factoring the tree into its arcs, and parsing is performed by searching for the highest scoring tree (Eisner, 1996; McDonald et al., 2005b). Transition-based parsers model the sequence of decisions of a shift-reduce parser, given previous decisions and current state, and parsing is performed by greedily choosing the highest scoring transition out of each successive parsing state or by searching for the best sequence of transitions (Ratnaparkhi et al., 1994; Yamada and Matsumoto, 2003; Nivre et al., 2004; Sagae and Lavie, 2005; Hall et al., 2006).

Both approaches most commonly use linear models to assign scores to arcs or decisions, so that a score is a dot-product of a feature vector $\mathbf{f}$ and a learned weight vector $\mathbf{w}$.

In sum, these two lines of research use different approximations to achieve tractability. Transition-based approaches solve a sequence of *local* problems in sequence, sacrificing global optimality guarantees and possibly expressive power (Abney et al., 1999). Graph-based methods perform *global* inference using score factorizations that correspond to strong independence assumptions (discussed in

§2.1.2). Recently, Nivre and McDonald (2008) proposed *combining* a graph-based and a transition-based parser and have shown a significant improvement for several languages by letting one of the parsers "guide" the other. Our stacked formalism (to be described in §3) generalizes this approach.

2.1.2 Arc factorization

In the successful graph-based method of McDonald et al. (2005b), an *arc factorization* independence assumption is used to ensure tractability. This assumption forbids any feature that depends on two or more arcs, permitting only "arc-factored" features (i.e. features that depend only on a single candidate arc $a \in A_y$ and on the input sequence x). This induces a decomposition of the feature vector $\mathbf{f}(x, y)$ as:

$$\mathbf{f}(x, y) = \sum_{a \in A_y} \mathbf{f}_a(x).$$

Parsing amounts to solving $\arg\max_{y \in \mathcal{Y}(x)} \mathbf{w}^\top \mathbf{f}(x, y)$, where $\mathbf{w}$ is a weight vector. With a projectivity constraint and arc factorization, the parsing problem can be solved in cubic time by dynamic programming (Eisner, 1996), and with a weaker "tree" constraint (permitting nonprojective parses) and arc factorization, a quadratic-time algorithm exists (Chu and Liu, 1965; Edmonds, 1967), as shown by McDonald et al. (2005b). In the projective case, the arc-factored assumption can be weakened in certain ways while maintaining polynomial parser runtime (Eisner and Satta, 1999), but not in the nonprojective case (McDonald and Satta, 2007), where finding the highest-scoring tree becomes NP-hard.

McDonald and Pereira (2006) adopted an approximation based on $O(n^3)$ projective parsing followed by rearrangement to permit crossing arcs, achieving higher performance. In §3 we adopt a framework that maintains $O(n^2)$ runtime (still exploiting the Chu-Liu-Edmonds algorithm) while approximating non arc-factored features.

2.2 Stacked Learning

Stacked generalization was first proposed by Wolpert (1992) and Breiman (1996) for regression. The idea is to include two "levels" of predictors. The first level, "level 0," includes one or more predictors $g_1, \ldots, g_K : \mathbb{R}^d \to \mathbb{R}$; each receives input $\mathbf{x} \in \mathbb{R}^d$ and outputs a prediction $g_k(\mathbf{x})$. The second level, "level 1," consists of a single function $h : \mathbb{R}^{d+K} \to \mathbb{R}$ that takes as input $\langle \mathbf{x}, g_1(\mathbf{x}), \ldots g_K(\mathbf{x}) \rangle$ and outputs a final prediction $\hat{y} = h(\mathbf{x}, g_1(\mathbf{x}), \ldots g_K(\mathbf{x}))$. The predictor, then, combines an ensemble (the g_k) with a meta-predictor (h).

Training is done as follows: the training data are split into L partitions, and L instances of the level 0 predictor are trained in a "leave-one-out" basis. Then, an augmented dataset is formed by letting each instance output predictions for the partition that was left out. Finally, each level 0 predictor is trained using the original dataset, and the level 1 predictor is trained on the augmented dataset, simulating the test-time setting when h is applied to a new instance $\mathbf{x}$ concatenated with $\langle g_k(\mathbf{x}) \rangle_k$.

This framework has also been applied to classification, for example with structured data. Some applications (including here) use only one classifier at level 0; recent work includes sequence labeling (Cohen and de Carvalho, 2005) and inference in conditional random fields (Kou and Cohen, 2007). Stacking is also intuitively related to transformation-based learning (Brill, 1993).

3 Stacked Dependency Parsing

We next describe how to use stacked learning for efficient, rich-featured dependency parsing.

3.1 Architecture

The architecture consists of two levels. At level 0 we include a single dependency parser. At runtime, this "level 0 parser" g processes an input sentence x and outputs the set of predicted edges that make up its estimation of the dependency tree, $\hat{y}_0 = g(x)$. At level 1, we apply a dependency parser—in this work, always a graph-based dependency parser—that uses basic factored features *plus* new ones from the edges *predicted* by the level 0 parser. The final parser predicts parse trees as $h(x, g(x))$, so that the total runtime is additive in calculating $h(\cdot)$ and $g(\cdot)$.

The stacking framework is agnostic about the form of g and h and the methods used to learn them from data. In this work we use two well-known, publicly available dependency parsers, MSTParser (McDonald et al., 2005b),[1] which implements ex-

[1] `http://sourceforge.net/projects/mstparser`

act first-order arc-factored nonprojective parsing (§2.1.2) and approximate second-order nonprojective parsing, and MaltParser (Nivre et al., 2006), which is a state-of-the-art transition-based parser.[2] We do not alter the training algorithms used in prior work for learning these two parsers from data. Using the existing parsers as starting points, we will combine them in a variety of ways.

3.2 Training

Regardless of our choices for the specific parsers and learning algorithms at level 0 and level 1, training is done as sketched in §2.2. Let $\mathcal{D}$ be a set of training examples $\{\langle x_i, y_i \rangle\}_i$.

1. Split training data $\mathcal{D}$ into L partitions $\mathcal{D}^1, \ldots, \mathcal{D}^L$.
2. Train L instances of the level 0 parser in the following way: the l-th instance, g^l, is trained on $\mathcal{D}^{-l} = \mathcal{D} \setminus \mathcal{D}^l$. Then use g^l to output predictions for the (unseen) partition $\mathcal{D}^l$. At the end, an augmented dataset $\tilde{\mathcal{D}} = \bigcup_{l=1}^{L} \tilde{\mathcal{D}}^l$ is built, so that $\tilde{\mathcal{D}} = \{\langle x_i, g(x_i), y_i \rangle\}_i$.
3. Train the level 0 parser g on the original training data $\mathcal{D}$.
4. Train the level 1 parser h on the augmented training data $\tilde{\mathcal{D}}$.

The runtime of this algorithm is $O(LT_0 + T_1)$, where T_0 and T_1 are the individual runtimes required for training level 0 and level 1 alone, respectively.

4 Two Views of Stacked Parsing

We next describe two motivations for stacking parsers: as a way of augmenting the features of a graph-based dependency parser or as a way to approximate higher-order models.

4.1 Adding Input Features

Suppose that the level 1 classifier is an arc-factored graph-based parser. The feature vectors will take the form[3]

$$\begin{aligned} \mathbf{f}(x, y) &= \mathbf{f}_1(x, y) \smile \mathbf{f}_2(x, \hat{y}_0, y) \\ &= \sum_{a \in A_y} \mathbf{f}_{1,a}(x) \smile \mathbf{f}_{2,a}(x, g(x)), \end{aligned}$$

where $\mathbf{f}_1(x, y) = \sum_{a \in A_y} \mathbf{f}_{1,a}(x)$ are regular arc-factored features, and $\mathbf{f}_2(x, \hat{y}_0, y) = \sum_{a \in A_y} \mathbf{f}_{2,a}(x, g(x))$ are the *stacked features*. An example of a stacked feature is a binary feature $f_{2,a}(x, g(x))$ that fires if and only if the arc a was predicted by g, i.e., if $a \in A_{g(x)}$; such a feature was used by Nivre and McDonald (2008).

It is difficult in general to decide whether the inclusion of such a feature yields a better parser, since features strongly correlate with each other. However, a popular heuristic for feature selection consists of measuring the *information gain* provided by each individual feature. In this case, we may obtain a closed-form expression for the information gain that $f_{2,a}(x, g(x))$ provides about the existence or not of the arc a in the actual dependency tree y. Let A and A' be binary random variables associated with the events $a \in A_y$ and $a' \in A_{g(x)}$, respectively. We have:

$$\begin{aligned} I(A; A') &= \sum_{a, a' \in \{0,1\}} p(a, a') \log_2 \frac{p(a, a')}{p(a)p(a')} \\ &= H(A') - \sum_{a \in \{0,1\}} p(a) H(A' | A = a). \end{aligned}$$

Assuming, for simplicity, that at level 0 the probability of false positives equals the probability of false negatives (i.e., $P_{\text{err}} \triangleq p(a' = 0 | a = 1) = p(a' = 1 | a = 0)$), and that the probability of true positives equals the probability of true negatives ($1 - P_{\text{err}} = p(a' = 0 | a = 0) = p(a' = 1 | a = 1)$), the expression above reduces to:

$$\begin{aligned} I(A; A') &= H(A') + P_{\text{err}} \log_2 P_{\text{err}} \\ &+ (1 - P_{\text{err}}) \log_2 (1 - P_{\text{err}}) \\ &= H(A') - H_{\text{err}}, \end{aligned}$$

where H_{err} denotes the entropy of the probability of error on each arc's prediction by the level 0 classifier. If $P_{\text{err}} \leq 0.5$ (i.e. if the level 0 classifier is better than random), then the information gain provided by this simple stacked feature increases with (a) the accuracy of the level 0 classifier, and (b) the entropy $H(A')$ of the distribution associated with its arc predictions.

4.2 Approximating Non-factored Features

Another way of interpreting the stacking framework is as a means to approximate a higher order model,

[2] `http://w3.msi.vxu.se/~jha/maltparser`

[3] We use $\smile$ to denote vector concatenation.

such as one that is not arc-factored, by using stacked features that make use of the predicted structure *around* a candidate arc. Consider a second-order model where the features decompose by arc and by arc *pair*:

$$\mathbf{f}(x,y) = \sum_{a_1 \in A_y} \left(\mathbf{f}_{a_1}(x) \smile \sum_{a_2 \in A_y} \mathbf{f}_{a_1,a_2}(x) \right).$$

Exact parsing under such model, with arbitrary second-order features, is intractable (McDonald and Satta, 2007). Let us now consider a stacked model in which the level 0 predictor outputs a parse $\hat{y}$. At level 1, we use arc-factored features that may be written as

$$\tilde{\mathbf{f}}(x,y) = \sum_{a_1 \in A_y} \left(\mathbf{f}_{a_1}(x) \smile \sum_{a_2 \in A_{\hat{y}}} \mathbf{f}_{a_1,a_2}(x) \right);$$

this model differs from the previous one only by replacing A_y by $A_{\hat{y}}$ in the index set of the second summation. Since $\hat{y}$ is given, this makes the latter model arc-factored, and therefore, tractable. We can now view $\tilde{\mathbf{f}}(x,y)$ as an approximation of $\mathbf{f}(x,y)$; indeed, we can bound the *score approximation error*,

$$\Delta s(x,y) = \left| \tilde{\mathbf{w}}^\top \tilde{\mathbf{f}}(x,y) - \mathbf{w}^\top \mathbf{f}(x,y) \right|,$$

where $\tilde{\mathbf{w}}$ and $\mathbf{w}$ stand respectively for the parameters learned for the stacked model and those that would be learned for the (intractable) exact second order model. We can bound $\Delta s(x,y)$ by spliting it into two terms: $\Delta s(x,y) =$

$$\begin{aligned} &\left| (\tilde{\mathbf{w}} - \mathbf{w})^\top \tilde{\mathbf{f}}(x,y) + \mathbf{w}^\top (\tilde{\mathbf{f}}(x,y) - \mathbf{f}(x,y)) \right| \\ &\leq \underbrace{\left| (\tilde{\mathbf{w}} - \mathbf{w})^\top \tilde{\mathbf{f}}(x,y) \right|}_{\triangleq \Delta s_{\text{tr}}(x,y)} + \underbrace{\left| \mathbf{w}^\top (\tilde{\mathbf{f}}(x,y) - \mathbf{f}(x,y)) \right|}_{\triangleq \Delta s_{\text{dec}}(x,y)}; \end{aligned}$$

where we introduced the terms Δs_{tr} and Δs_{dec} that reflect the portion of the score approximation error that are due to *training error* (i.e., different parameterizations of the exact and approximate models) and *decoding error* (same parameterizations, but different feature vectors). Using Hölder's inequality, the former term can be bounded as:

$$\begin{aligned} \Delta s_{\text{tr}}(x,y) &= \left| (\tilde{\mathbf{w}} - \mathbf{w})^\top \tilde{\mathbf{f}}(x,y) \right| \\ &\leq \|\tilde{\mathbf{w}} - \mathbf{w}\|_1 \cdot \|\tilde{\mathbf{f}}(x,y)\|_\infty \\ &\leq \|\tilde{\mathbf{w}} - \mathbf{w}\|_1; \end{aligned}$$

where $\|.\|_1$ and $\|.\|_\infty$ denote the ℓ_1-norm and sup-norm, respectively, and the last inequality holds when the features are binary (so that $\|\tilde{\mathbf{f}}(x,y)\|_\infty \leq 1$). The proper way to bound the term $\|\tilde{\mathbf{w}} - \mathbf{w}\|_1$ depends on the training algorithm. As for the decoding error term, it can bounded for a given weight vector $\mathbf{w}$, sentence x, candidate tree y, and level 0 prediction $\hat{y}$. Decomposing the weighted vector as $\mathbf{w} = \mathbf{w}_1 \smile \mathbf{w}_2$, $\mathbf{w}_2$ being the sub-vector associated with the second-order features, we have respectively: $\Delta s_{\text{dec}}(x,y) =$

$$\begin{aligned} &\left| \mathbf{w}^\top (\tilde{\mathbf{f}}(x,y) - \mathbf{f}(x,y)) \right| \\ &= \left| \sum_{a_1 \in A_y} \mathbf{w}_2^\top \left(\sum_{a_2 \in A_{\hat{y}}} \mathbf{f}_{a_1,a_2}(x) - \sum_{a_2 \in A_y} \mathbf{f}_{a_1,a_2}(x) \right) \right| \\ &\leq \sum_{a_1 \in A_y} \sum_{a_2 \in A_{\hat{y}} \Delta A_y} \left| \mathbf{w}_2^\top \mathbf{f}_{a_1,a_2}(x) \right| \\ &\leq \sum_{a_1 \in A_y} |A_{\hat{y}} \Delta A_y| \cdot \max_{a_2 \in A_{\hat{y}} \Delta A_y} \left| \mathbf{w}_2^\top \mathbf{f}_{a_1,a_2}(x) \right| \\ &= \sum_{a_1 \in A_y} 2L(y,\hat{y}) \cdot \max_{a_2 \in A_{\hat{y}} \Delta A_y} \left| \mathbf{w}_2^\top \mathbf{f}_{a_1,a_2}(x) \right|, \end{aligned}$$

where $A_{\hat{y}} \Delta A_y \triangleq (A_{\hat{y}} - A_y) \cup (A_y - A_{\hat{y}})$ denotes the *symmetric difference* of the sets $A_{\hat{y}}$ and A_y, which has cardinality $2L(y,\hat{y})$, *i.e.*, twice the Hamming distance between the sequences of heads that characterize y and the predicted parse $\hat{y}$. Using Hölder's inequality, we have both

$$\begin{aligned} \left| \mathbf{w}_2^\top \mathbf{f}_{a_1,a_2}(x) \right| &\leq \|\mathbf{w}_2\|_1 \cdot \|\mathbf{f}_{a_1,a_2}(x)\|_\infty \\ \text{and } \left| \mathbf{w}_2^\top \mathbf{f}_{a_1,a_2}(x) \right| &\leq \|\mathbf{w}_2\|_\infty \cdot \|\mathbf{f}_{a_1,a_2}(x)\|_1. \end{aligned}$$

Assuming that all features are binary valued, we have that $\|\mathbf{f}_{a_1,a_2}(x)\|_\infty \leq 1$ and that $\|\mathbf{f}_{a_1,a_2}(x)\|_1 \leq N_{f,2}$, where $N_{f,2}$ denotes the maximum number of active second order features for any possible pair of arcs (a_1, a_2). Therefore:

$$\Delta s_{\text{dec}}(x,y) \leq 2nL(y,\hat{y}) \min\{\|\mathbf{w}_2\|_1, N_{f,2} \cdot \|\mathbf{w}_2\|_\infty\},$$

where n is the sentence length. Although this bound can be loose, it suggests (intuitively) that the score approximation degrades as the predicted tree $\hat{y}$ gets farther away from the true tree y (in Hamming distance). It also degrades with the magnitude of weights associated with the second-order features,

Name	Description
PredEdge	Indicates whether the candidate edge was present, and what was its label.
Sibling	Lemma, POS, link label, distance and direction of attachment of the previous and and next predicted siblings
GrandParents	Lemma, POS, link label, distance and direction of attachment of the grandparent of the current modifier
PredHead	Predicted head of the candidate modifier (if PredEdge=0)
AllChildren	Sequence of POS and link labels of all the predicted children of the candidate head

Table 1: Feature sets derived from the level 0 parser.

Subset	Description
A	PredEdge
B	PredEdge+Sibling
C	PredEdge+Sibling+GrandParents
D	PredEdge+Sibling+GrandParents+PredHead
E	PredEdge+Sibling+GrandParents+PredHead+ AllChildren

Table 2: Combinations of features enumerated in Table 1 used for stacking. A is a replication of (Nivre and McDonald, 2008), except for the modifications described in footnote 4.

which suggests that a separate regularization of the first-order and stacked features might be beneficial in a stacking framework.

As a side note, if we set each component of the weight vector to one, we obtain a bound on the ℓ_1-norm of the feature vector difference, $\left\|\tilde{\mathbf{f}}(x,y) - \mathbf{f}(x,y)\right\|_1 \leq 2nL(y,\hat{y})N_{f,2}$.

5 Experiments

In the following experiments we demonstrate the effectiveness of stacking parsers. As noted in §3.1, we make use of two component parsers, the graph-based MSTParser and the transition-based MaltParser.

5.1 Implementation and Experimental Details

The publicly available version of MSTParser performs parsing and labeling jointly. We adapted this system to first perform unlabeled parsing, then label the arcs using a log-linear classifier with access to the full unlabeled parse (McDonald et al., 2005a; McDonald et al., 2005b; McDonald and Pereira, 2006). In stacking experiments, the arc labels from the level 0 parser are also used as a feature.[4]

In the following subsections, we refer to our modification of the MSTParser as MST_{1O} (the arc-factored version) and MST_{2O} (the second-order arc-pair-factored version). All our experiments use the non-projective version of this parser. We refer to the MaltParser as $Malt$.

We report experiments on twelve languages from the CoNLL-X shared task (Buchholz and Marsi, 2006).[5] All experiments are evaluated using the *labeled attachment score* (LAS), using the default settings.[6] Statistical significance is measured using Dan Bikel's randomized parsing evaluation comparator with 10,000 iterations.[7] The additional features used in the level 1 parser are enumerated in Table 1 and their various subsets are depicted in Table 2. The PredEdge features are exactly the six features used by Nivre and McDonald (2008) in their **MST**$_{\text{Malt}}$ parser; therefore, feature set A is a replication of this parser except for modifications noted in footnote 4. In all our experiments, the number of partitions used to create $\tilde{\mathcal{D}}$ is $L = 2$.

5.2 Experiment: $MST_{2O} + MST_{2O}$

Our first experiment stacks the highly accurate MST_{2O} parser with itself. At level 0, the parser uses only the standard features (§5.1), and at level 1, these are augmented by various subsets of features of x along with the output of the level 0 parser, $g(x)$ (Table 2). The results are shown in Table 3. While we see improvements over the single-parser baseline

[4]We made other modifications to MSTParser, implementing many of the successes described by (McDonald et al., 2006). Our version of the code is publicly available at `http://www.ark.cs.cmu.edu/MSTParserStacked`. The modifications included an approximation to lemmas for datasets without lemmas (three-character prefixes), and replacing morphology/word and morphology/lemma features with morphology/POS features.

[5]The CoNLL-X shared task actually involves thirteen languages; our experiments do not include Czech (the largest dataset), due to time constraints. Therefore, the average results plotted in the last rows of Tables 3, 4, and 5 are not directly comparable with previously published averages over thirteen languages.

[6]`http://nextens.uvt.nl/~conll/software.html`

[7]`http://www.cis.upenn.edu/~dbikel/software.html`

	MST_{2O}	$+MST_{2O}$, A	$+MST_{2O}$, B	$+MST_{2O}$, C	$+MST_{2O}$, D	$+MST_{2O}$, E
Arabic	67.88	66.91	67.41	67.68	67.37	**68.02**
Bulgarian	87.31	87.39	87.03	**87.61**	87.57	87.55
Chinese	**87.57**	87.16	87.24	87.48	87.42	87.48
Danish	85.27	85.39	**85.61**	85.57	85.43	85.57
Dutch	79.99	79.79	79.79	79.83	**80.17**	80.13
German	**87.44**	86.92	87.32	87.32	87.26	87.04
Japanese	90.93	**91.41**	91.21	91.35	91.11	91.19
Portuguese	87.12	**87.26**	86.88	87.02	87.04	86.98
Slovene	74.02	**74.30**	**74.30**	74.00	74.14	73.94
Spanish	82.43	82.17	82.35	**82.81**	82.53	82.75
Swedish	82.87	82.99	82.95	82.51	**83.01**	82.69
Turkish	**60.11**	59.47	59.25	59.47	59.45	59.31
Average	**81.08**	80.93	80.94	81.05	81.04	81.05

Table 3: Results of stacking MST_{2O} with itself at both level 0 and level 1. Column 2 enumerates LAS for MST_{2O}. Columns 3–6 enumerate results for four different stacked feature subsets. Bold indicates best results for a particular language.

for nine languages, the improvements are small (less than 0.5%). One of the biggest concerns about this model is the fact that it stacks two predictors that are very similar in nature: both are graph-based and share the features $\mathbf{f}_{1,a}(x)$. It has been pointed out by Breiman (1996), among others, that the success of ensemble methods like stacked learning strongly depends on how uncorrelated the individual decisions made by each predictor are from the others' decisions.[8] This experiment provides further evidence for the claim.

5.3 Experiment: $Malt + MST_{2O}$

We next use MaltParser at level 0 and the *second-order* arc-pair-factored MST_{2O} at level 1. This extends the experiments of Nivre and McDonald (2008), replicated in our feature subset A.

Table 4 enumerates the results. Note that the best-performing stacked configuration for each and every language outperforms MST_{2O}, corroborating results reported by Nivre and McDonald (2008). The best performing stacked configuration outperforms $Malt$ as well, except for Japanese and Turkish. Further, our non-arc-factored features largely outperform subset A, except on Bulgarian, Chinese, and Japanese. On average, the best feature configuration is E, which is statistically significant over $Malt$ and MST_{2O} with $p < 0.0001$, and over feature subset A with $p < 0.01$.

5.4 Experiment: $Malt + MST_{1O}$

Finally, we consider stacking MaltParser with the first-order, arc-factored MSTParser. We view this approach as perhaps the most promising, since it is an exact parsing method with the quadratic runtime complexity of MST_{1O}.

Table 5 enumerates the results. For all twelve languages, some stacked configuration outperforms MST_{1O} and also, surprisingly, MST_{2O}, the second order model. This provides empirical evidence that using rich features from MaltParser at level 0, a stacked level 1 first-order MSTParser can outperform the second-order MSTParser.[9] In only two cases (Japanese and Turkish), the MaltParser slightly outperforms the stacked parser.

On average, feature configuration D performs the best, and is statistically significant over $Malt$, MST_{1O}, and MST_{2O} with $p < 0.0001$, and over feature subset A with $p < 0.05$. Encouragingly, this configuration is barely outperformed by configura-

[8]This claim has a parallel in the cotraining method (Blum and Mitchell, 1998), whose performance is bounded by the degree of independence between the two feature sets.

[9]Recall that MST_{2O} uses approximate *search*, as opposed to stacking, which uses approximate *features*.

	Malt	MST_{2O}	$Malt + MST_{2O}$	$Malt + MST_{2O}$, A	$Malt + MST_{2O}$, B	$Malt + MST_{2O}$, C	$Malt + MST_{2O}$, D	$Malt + MST_{2O}$, E
Arabic	66.71	67.88	68.56	**69.12**	68.64	68.34	68.92	
Bulgarian	87.41	87.31	**88.99**	88.89	88.89	88.93	88.91	
Chinese	86.92	87.57	**88.41**	88.31	88.29	88.13	**88.41**	
Danish	84.77	85.27	86.45	86.67	**86.79**	86.13	86.71	
Dutch	78.59	79.99	80.75	81.47	81.47	**81.51**	81.29	
German	85.82	87.44	88.16	88.50	88.56	**88.68**	88.38	
Japanese	**91.65**	90.93	91.63	91.43	91.59	91.61	91.49	
Portuguese	87.60	87.12	88.00	88.24	**88.30**	88.18	88.22	
Slovene	70.30	74.02	76.62	76.00	76.60	76.18	**76.72**	
Spanish	81.29	82.43	83.09	**83.73**	83.47	83.21	83.43	
Swedish	84.58	82.87	84.92	84.60	84.80	**85.16**	84.88	
Turkish	**65.68**	60.11	64.35	64.51	64.51	65.07	65.21	
Average	80.94	81.08	82.52	82.58	82.65	82.59	**82.71**	

Table 4: Results of stacking *Malt* and MST_{2O} at level 0 and level 1, respectively. Columns 2–4 enumerate LAS for *Malt*, MST_{2O} and $Malt + MST_{2O}$ as in Nivre and McDonald (2008). Columns 5–8 enumerate results for four other stacked feature configurations. Bold indicates best result for a language.

tion A of $Malt + MST_{2O}$ (see Table 4), the difference being statistically insignificant ($p > 0.05$). This shows that stacking *Malt* with the exact, arc-factored MST_{1O} bridges the difference between the individual MST_{1O} and MST_{2O} models, by approximating higher order features, but maintaining an $O(n^2)$ runtime and finding the model-optimal parse.

5.5 Disagreement as a Confidence Measure

In pipelines or semisupervised settings, it is useful when a parser can provide a *confidence* measure alongside its predicted parse tree. Because stacked predictors use ensembles with observable outputs, differences among those outputs may be used to estimate confidence in the final output. In stacked dependency parsing, this can be done (for example) by measuring the Hamming distance between the outputs of the level 0 and 1 parsers, $L(g(x), h(x))$. Indeed, the bound derived in §4.2 suggests that the second-order approximation degrades for candidate parses y that are Hamming-far from $g(x)$; therefore, if $L(g(x), h(x))$ is large, the best score $s(x, h(x))$ may well be "biased" due to misleading neighboring information provided by the level 0 parser.

We illustrate this point with an empirical analysis of the level 0/1 disagreement for the set of experiments described in §5.3; namely, we compare the

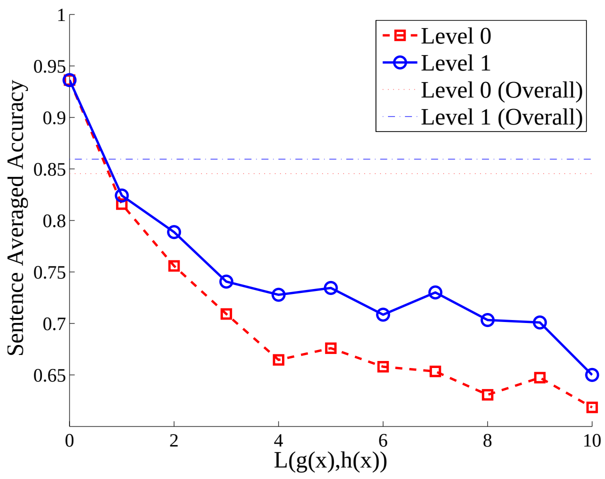


Figure 2: Accuracy as a function of token disagreement between level 0 and level 1. The x-axis is the Hamming distance $L(g(x), h(x))$, i.e., the number of tokens where level 0 and level 1 disagree. The y-axis is the accuracy averaged over sentences that have the specified Hamming distance, both for level 0 and level 1.

	Malt	MST_{1O}	MST_{2O}	$Malt + MST_{1O}$, A	$Malt + MST_{1O}$, B	$Malt + MST_{1O}$, C	$Malt + MST_{1O}$, D	$Malt + MST_{1O}$, E
Arabic	66.71	66.81	67.88	68.40	68.50	68.20	68.42	**68.68**
Bulgarian	87.41	86.65	87.31	88.55	88.67	88.75	88.71	**88.79**
Chinese	86.92	86.60	87.57	87.67	87.73	**87.83**	87.67	87.61
Danish	84.77	84.87	85.27	**86.59**	86.27	86.21	86.35	86.15
Dutch	78.59	78.95	79.99	80.53	81.51	80.71	**81.61**	81.37
German	85.82	86.26	87.44	88.18	88.30	88.20	88.36	**88.42**
Japanese	**91.65**	91.01	90.93	91.55	91.53	91.51	91.43	91.57
Portuguese	87.60	86.28	87.12	88.16	88.26	**88.46**	88.26	88.36
Slovene	70.30	73.96	74.02	75.84	75.64	75.42	**75.96**	75.64
Spanish	81.29	81.07	82.43	82.61	**83.13**	**83.13**	83.09	82.99
Swedish	84.58	81.88	82.87	**84.86**	84.62	84.64	84.82	84.76
Turkish	**65.68**	59.63	60.11	64.49	64.97	64.47	64.63	64.61
Average	80.94	80.33	81.08	82.28	82.42	82.29	**82.44**	82.41

Table 5: Results of stacking $Malt$ and MST_{1O} at level 0 and level 1, respectively. Columns 2–4 enumerate LAS for $Malt$, MST_{1O} and MST_{2O}. Columns 5–9 enumerate results for five different stacked feature configurations. Bold indicates the best result for a language.

level 0 and level 1 predictions under the best overall configuration (configuration E of $Malt + MST_{2O}$). Figure 2 depicts accuracy as a function of level 0-level 1 disagreement (in number of tokens), averaged over all datasets.

We can see that performance degrades steeply when the disagreement between levels 0 and 1 increases in the range 0–4, and then behaves more irregularly but keeping the same trend. This suggests that the Hamming distance $L(g(x), h(x))$ is informative about parser performance and may be used as a confidence measure.

6 Conclusion

In this work, we made use of stacked learning to improve dependency parsing. We considered an architecture with two layers, where the output of a standard parser in the first level provides new features for a parser in the subsequent level. During learning, the second parser learns to correct mistakes made by the first one. The novelty of our approach is in the exploitation of higher-order predicted edges to simulate non-local features in the second parser. We provided a novel interpretation of stacking as feature approximation, and our experimental results show rich-featured stacked parsers outperforming state-of-the-art single-layer and ensemble parsers. Notably, using a simple arc-factored parser at level 1, we obtain an exact $O(n^2)$ stacked parser that outperforms earlier approximate methods (McDonald and Pereira, 2006).

Acknowledgments

The authors thank the anonymous reviewers for helpful comments, Vitor Carvalho, William Cohen, and David Smith for interesting discussions, and Ryan McDonald and Joakim Nivre for providing us their code and preprocessed datasets. A.M. was supported by a grant from FCT through the CMU-Portugal Program and the Information and Communications Technologies Institute (ICTI) at CMU. N.S. was supported by NSF IIS-0713265 and an IBM faculty award. E.X. was supported by NSF DBI-0546594, DBI-0640543, and IIS-0713379.

References

S. P. Abney, D. A. McAllester, and F. Pereira. 1999. Relating probabilistic grammars and automata. In *Proceedings of ACL*.

A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of COLT*.

L. Breiman. 1996. Stacked regressions. *Machine Learning*, 24:49.

E. Brill. 1993. *A Corpus-Based Approach to Language Learning*. Ph.D. thesis, University of Pennsylvania.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL*.

Y. J. Chu and T. H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

W. W. Cohen and V. Rocha de Carvalho. 2005. Stacked sequential learning. In *Proceedings of IJCAI*.

A. Culotta and J. Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of ACL*.

Y. Ding and M. Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammar. In *Proceedings of ACL*.

J. Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

J. Eisner and G. Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of ACL*.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING*.

J. Hall, J. Nivre, and J. Nilsson. 2006. Discriminative classifiers for deterministic dependency parsing. In *Proceedings of ACL*.

Z. Kou and W. W. Cohen. 2007. Stacked graphical models for efficient inference in Markov random fields. In *Proceedings of SDM*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

P. Liang, H. Daumé, and D. Klein. 2008. Structure compilation: trading structure for features. In *Proceedings of ICML*.

R. McDonald and J. Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of EMNLP-CoNLL*.

R. T. McDonald and F. C. N. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*.

R. McDonald and G. Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of IWPT*.

R. McDonald, K. Crammer, and F. Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of ACL*.

R. T. McDonald, F. Pereira, K. Ribarov, and J. Hajic. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT-EMNLP*.

R. McDonald, K. Lerman, and F. Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings CoNLL*.

J. Nivre and R. McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-HLT*.

J. Nivre, J. Hall, and J. Nilsson. 2004. Memory-based dependency parsing. In *Proceedings of CoNLL*.

J. Nivre, J. Hall, J. Nilsson, G. Eryiğit, and S. Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of CoNLL*.

A. Ratnaparkhi, S. Roukos, and R. T. Ward. 1994. A maximum entropy model for parsing. In *Proceedings of ICSLP*.

S. Riedel and J. Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *Proceedings of EMNLP*.

K. Sagae and A. Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of IWPT*.

D. A. Smith and J. Eisner. 2008. Dependency parsing by belief propagation. In *Proceedings of EMNLP*.

I. Titov and J. Henderson. 2007. A latent variable model for generative dependency parsing. In *Proceedings of IWPT*.

M. Wang, N. A. Smith, and T. Mitamura. 2007. What is the Jeopardy model? A quasi-synchronous grammar for QA. In *Proceedings of EMNLP-CoNLL*.

D. Wolpert. 1992. Stacked generalization. *Neural Networks*, 5(2):241–260.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*.

Better Binarization for the CKY Parsing

Xinying Song† [*] **Shilin Ding**§ [*] **Chin-Yew Lin**‡
†MOE-MS Key Laboratory of NLP and Speech, Harbin Institute of Technology, Harbin, China
§Department of Statistics, University of Wisconsin-Madison, Madison, USA
‡Microsoft Research Asia, Beijing, China
xysong@mtlab.hit.edu.cn dingsl@gmail.com cyl@microsoft.com

Abstract

We present a study on how grammar binarization empirically affects the efficiency of the CKY parsing. We argue that binarizations affect parsing efficiency primarily by affecting the number of incomplete constituents generated, and the effectiveness of binarization also depends on the nature of the input. We propose a novel binarization method utilizing rich information learnt from training corpus. Experimental results not only show that different binarizations have great impacts on parsing efficiency, but also confirm that our learnt binarization outperforms other existing methods. Furthermore we show that it is feasible to combine existing parsing speed-up techniques with our binarization to achieve even better performance.

1 Introduction

Binarization, which transforms an n-ary grammar into an equivalent binary grammar, is essential for achieving an $O(n^3)$ time complexity in the context-free grammar parsing. $O(n^3)$ tabular parsing algorithms, such as the CKY algorithm (Kasami, 1965; Younger, 1967), the GHR parser (Graham et al., 1980), the Earley algorithm (Earley, 1970) and the chart parsing algorithm (Kay, 1980; Klein and Manning, 2001) all convert their grammars into binary branching forms, either explicitly or implicitly (Charniak et al., 1998).

In fact, the number of all possible binarizations of a production with $n+1$ symbols on its right hand side is known to be the nth *Catalan Number* $C_n = \frac{1}{n+1}\binom{2n}{n}$. All binarizations lead to the same parsing accuracy, but maybe different parsing efficiency, i.e. parsing speed. We are interested in investigating whether and how binarizations will affect the efficiency of the CKY parsing.

Do different binarizations lead to different parsing efficiency? Figure 1 gives an example to help answer this question. Figure 1(a) illustrates the correct parse of the phrase "*get the bag and go*". We assume that $NP \rightarrow NP\ CC\ NP$ is in the original grammar. The symbols enclosed in square brackets in the figure are intermediate symbols.

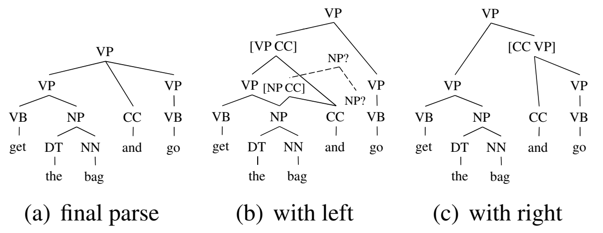


Figure 1: Parsing with left and right binarization.

If a left binarized grammar is used, see Figure 1(b), an extra constituent $[NP\,CC]$ spanning "*the bag and*" will be produced. Because rule $[NP\,CC] \rightarrow NP\,CC$ is in the left binarized grammar and there is an NP over "*the bag*" and a CC over the right adjacent "*and*". Having this constituent is unnecessary, because it lacks an NP to the right to complete the production. However, if a right binarization is used, as shown in Figure 1(c), such unnecessary constituent can be avoided.

One observation from this example is that different binarizations affect constituent generation, thus affect parsing efficiency. Another observation is that

[*]This work was done when Xinying Song and Shilin Ding were visiting students at Microsoft Research Asia.

for rules like $X \to Y\,C\,C\,Y$, it is more suitable to binarize them in a right branching way. This can be seen as a linguistic nature: for "*and*", usually the right neighbouring word can indicate the correct parse. A good binarization should reflect such liguistic nature.

In this paper, we aim to study the effect of binarization on the efficiency of the CKY parsing. To our knowledge, this is the first work on this problem.

We propose the problem to find the optimal binarization in terms of parsing efficiency (Section 3). We argue that binarizations affect parsing efficiency primarily by affecting the number of incomplete constituents generated, and the effectiveness of binarization also depends on the nature of the input (Section 4). Therefore we propose a novel binarization method utilizing rich information learnt from training corpus (Section 5). Experimental results show that our binarization outperforms other existing methods (Section 7.2).

Since binarization is usually a preprocessing step before parsing, we argue that better performance can be achieved by combining other parsing speed-up techniques with our binarization (Section 6). We conduct experiments to confirm this (Section 7.3).

2 Binarization

In this paper we assume that the *original* grammar, perhaps after preprocessing, contains no ϵ-productions or useless symbols. However, we allow the existence of unary productions, since we adopt an extended version of the CKY algorithm which can handle the unary productions. Moreover we do not distinguish nonterminals and terminals explicitly. We treat them as symbols. What we focus on is the procedure of binarization.

Definition 1. A *binarization* is a function π, mapping an n-ary grammar G to an equivalent binary grammar G'. We say that G' is a binarized grammar of G, denoted as $\pi(G)$.

Two grammars are *equivalent* if they define the same probability distribution over strings (Charniak et al., 1998).

We use the most widely used *left binarization* (Aho and Ullman, 1972) to show the procedure of binarization, as illustrated in Table 1, where p and q are the probabilities of the productions.

Original grammar	Left binarized grammar
$Y \to A\,B\,C : p$	$[A\,B] \to A\,B : 1.0$
$Z \to A\,B\,D : q$	$Y \to [A\,B]\,C : p$
	$Z \to [A\,B]\,D : q$

Table 1: Left binarization

In the binarized grammar, symbols of form $[A\,B]$ are *new* (also called *intermediate*) nonterminals. Left binarization always selects the left most pair of symbols and combines them to form an intermediate nonterminal. This procedure is repeated until all productions are binary.

In this paper, we assume that all binarizations follow the fashion above, except that the choice of pair of symbols for combination can be arbitrary. Next we show three other known binarizations.

Right binarization is almost the same with left binarization, except that it always selects the right most pair, instead of left, to combine.

Head binarization always binarizes from the head outward (Klein and Manning, 2003b). Please refer to Charniak et al. (2006) for more details.

Compact binarization (Schmid, 2004) tries to minimize the size of the binarized grammar. It leads to a *compact* grammar. We therefore call it compact binarization. It is done via a greedy approach: it always selects the pair that occurs most on the right hand sides of rules to combine.

3 The optimal binarization

The optimal binarization should help CKY parsing to achieve its best efficiency. We formalize the idea as follows:

Definition 2. The *optimal binarization* is π^*, for a given n-ary grammar G and a test corpus C:

$$\pi^* = \arg\min_{\pi} T(\pi(G), C) \tag{1}$$

where $T(\pi(G), C)$ is the running time for CKY to parse corpus C, using the binarized grammar $\pi(G)$.

It is hard to find the optimal binarization directly from Definition 2. We next give an empirical analysis of the running time of the CKY algorithm and simplify the problem by introducing assumptions.

3.1 Analysis of CKY parsing efficiency

It is known that the complexity of the CKY algorithm is $O(n^3 L)$. The constant L depends on the binarized grammar in use. Therefore binarization will affect L. Our goal is to find a good binarization that makes parsing more efficient.

It is also known that in the inner most loop of CKY as shown in Algorithm 1, the for-statement in Line 1 can be implemented in several different methods. The choice will affect the efficiency of CKY. We present here four possible methods:

M1 Enumerate all rules $X \rightarrow Y\,Z$, and check if Y is in left span and Z in right span.

M2 For each Y in left span, enumerate all rules $X \rightarrow Y\,Z$, and check if Z is in right span.

M3 For each Z in right span, enumerate all rules $X \rightarrow Y\,Z$, and check if Y is in left span.

M4 Enumerate each Y in left span and Z in right span[1], check if there are any rules $X \rightarrow Y\,Z$.

Algorithm 1 The inner most loop of CKY

```
1: for $X \rightarrow Y Z$, $Y$ in left span and $Z$ in right span
2:     Add $X$ to parent span
```

3.2 Model assumption

We have shown that both binarization and the for-statement implementation in the inner most loop of CKY will affect the parsing speed.

About the for-statement implementations, no previous study has addressed which one is superior. The actual choice may affect our study on binarization. If using M1, since it enumerates all rules in the grammar, the optimal binarization will be the one with minimal number of rules, i.e. minimal binarized grammar size. However, M1 is usually not preferred in practice (Goodman, 1997). For other methods, it is hard to tell which binarization is optimal theoretically. In this paper, for simplicity reasons we do not consider the effect of for-statement implementations on the optimal binarization.

On the other hand, it is well known that reducing the number of constituents produced in parsing can greatly improve CKY parsing efficiency. That is how most thresholding systems (Goodman, 1997; Tsuruoka and Tsujii, 2004; Charniak et al., 2006) speed up CKY parsing. Apparently, the number of constituents produced in parsing is not affected by for-statement implementations.

Therefore we assume that the running time of CKY is primarily determined by the number of constituents generated in parsing. We simplify the optimal binarization to be:

$$\pi^* \approx \arg\min_{\pi} E(\pi(G), C) \qquad (2)$$

where $E(\pi(G), C)$ is the number of constituents generated when CKY parsing C with $\pi(G)$.

We next discuss how binarizations affect the number of constituents generated in parsing, and present our algorithm for finding a good binarization.

4 How binarizations affect constituents

Throughout this section and the next, we will use an example to help illustrate the idea. The grammar is:

$$\begin{array}{rl} X & \rightarrow A\ B\ C\ D \\ Y & \rightarrow A\ B\ C \\ C & \rightarrow C\ D \\ Z & \rightarrow A\ B\ C\ E \\ W & \rightarrow F\ C\ D\ E \end{array}$$

The input sentence is ${}_0A_1B_2C_3D_4E_5$, where the subscripts are used to indicate the positions of spans. For example, $[1,3]$ stands for $B\,C$. The final parse[2] is shown in Figure 2. Symbols surrounded by dashed circles are fictitious, which do not actually exist in the parse.

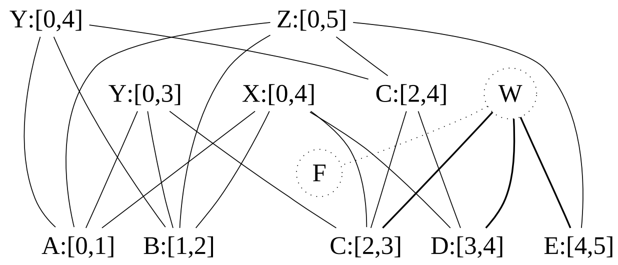


Figure 2: Parse of the sentence $A\,B\,C\,D\,E$

4.1 Complete and incomplete constituents

In the procedure of CKY parsing, there are two kinds of constituents generated: *complete* and *incomplete*.

Complete constituents (henceforth CCs) are those composed by the original grammar symbols and

[1]Note that we should skip Y (Z) if it never appears as the first (second) symbol on the right hand side of any rule.

[2]More precisely, it is more than a parse tree for it contains all symbols recognized in parsing.

spans. For example in Figure 2, $X:[0,4]$, $Y:[0,3]$ and Y:$[0,4]$ are all CCs.

Incomplete constituents (henceforth ICs) are those labeled by intermediate symbols. Figure 2 does not show them directly, but we can still read the possible ones. For example, if the binarized grammar in use contains an intermediate symbol $[A\,B\,C]$, then there will be two related ICs $[A\,B\,C]$:$[0,3]$ and $[A\,B\,C]$:$[0,4]$ (the latter is due to C:$[2,4]$) produced in parsing. ICs represent the intermediate steps to recognize and complete CCs.

4.2 Impact on complete constituents

Binarizations do not affect whether a CC will be produced. If there is a CC in the parse, whatever binarization we use, it will be produced. The difference merely lies on what intermediate ICs are used. Therefore given a grammar and an input sentence, no matter what binarization is used, the CKY parsing will generate the same set of CCs.

For example in Figure 2 there is a CC $X:[0,4]$, which is associated with rule $X \rightarrow A\,B\,C\,D$. No matter what binarization we use, this CC will be recognized eventually. For example if using left binarization, we will get $[A\,B]$:$[0,2]$, $[A\,B\,C]$:$[0,3]$ and finally $X:[0,4]$; if using right binarization, we will get $[C\,D]$:$[2,4]$, $[B\,C\,D]$:$[1,4]$ and again X:$[0,4]$.

4.3 Impact on incomplete constituents

Binarizations do affect the generation of ICs, because they generate different intermediate symbols. We discuss the impact on two aspects:

Shared IC. Some ICs can be used to generate multiple CCs in parsing. We call them *shared*. If a binarization can lead to more shared ICs, then overall there will be fewer ICs needed in parsing.

For example, in Figure 2, if we use left binarization, then $[A\,B]$:$[0,2]$ can be shared to generate both $X:[0,4]$ and $Y:[0,3]$, in which we can save one IC overall. However, if right binarization is used, there will be no common ICs to share in the generation steps of $X:[0,4]$ and $Y:[0,3]$, and overall there are one more IC generated.

Failed IC. For a CC, if it can be recognized eventually by applying an original rule of length k, whatever binarization to use, we will have to generate the same number of $k-2$ ICs before we can complete the CC. However, if the CC cannot be fully recognized but only partially recognized, then the number of ICs needed will be quite different.

For example, in Figure 2, the rule $W \rightarrow F\,C\,D\,E$ can be only partially recognized over $[2,5]$, so it cannot generate the corresponding CC. Right binarization needs two ICs ($[D\,E]$:$[3,5]$ and $[C\,D\,E]$:$[2,5]$) to find that the CC cannot be recognized, while left binarization needs none.

As mentioned earlier, ICs are auxiliary means to generate CCs. If an IC cannot help generate any CCs, it is totally useless and even harmful. We call such an IC *failed*, otherwise it is *successful*. Therefore, if a binarization can help generate fewer failed ICs then parsing would be more efficient.

4.4 Binarization and the nature of the input

Now we show that the impact of binarization also depends on the actual input. When the input changes, the impact may also change.

For example, in the previous example about the rule $W \rightarrow F\,C\,D\,E$ in Figure 2, we believe that left binarization is better based on the observation that there are more snippets of $[C\,D\,E]$ in the input which lack for F to the left. If there are more snippets of $[F\;C\;D]$ in the input lacking for E to the right, then right binarization would be better.

The discussion above confirms such a view: the effect of binarization depends on the nature of the input language, and a good binarization should reflect this nature. This accords with our intuition. So we use training corpus to learn a good binarization. And we verify the effectiveness of the learnt binarization using a test corpus with the same nature.

In summary, binarizations affect the efficiency of parsing primarily by affecting the number of ICs generated, where more shared and fewer failed ICs will help lead to higher efficiency. Meanwhile, the effectiveness of binarization also depends on the nature of its input language.

5 Towards a good binarization

Based on the analysis in the previous section, we employ a greedy approach to find a good binarization. We use training corpus to compute metrics for every possible intermediate symbol. We use this information to greedily select the best pair to combine.

5.1 Algorithm

Given the original grammar G and training corpus C, for every sentence in C, we firstly obtain the final parse (like Figure 2). For every possible intermediate symbol, i.e. every ngram of the original symbols, denoted by w, we compute the following two metrics:

1. How many ICs labeled by w can be generated in the final parse, denoted by $num(w)$ (number of related ICs).

2. How many CCs can be generated via ICs labeled by w, denoted by $ctr(w)$ (contribution of related ICs).

For example in Figure 2, for a possible intermediate symbol $[A\,B\,C]$, there are two related ICs ($[A\,B\,C] : [0, 3]$ and $[A\,B\,C] : [0, 4]$) in the parse, so we have $num([A\,B\,C]) = 2$. Meanwhile, four CCs ($Y : [0, 3]$, $X : [0, 4]$, $Y : [0, 4]$ and $Z : [0, 5]$) can be generated from the two related ICs. Therefore $ctr([A\,B\,C]) = 4$. We list the two metrics for every ngram in Figure 2 in Table 2. We will discuss how to compute these two metrics in Section 5.2.

w	num	ctr	w	num	ctr
$[A\,B]$	1	4	$[B\,C\,E]$	1	1
$[A\,B\,C]$	2	4	$[C\,D]$	1	2
$[A\,B\,C\,D]$	1	1	$[C\,D\,E]$	1	0
$[A\,B\,C\,E]$	1	1	$[C\,E]$	1	1
$[B\,C]$	2	4	$[D\,E]$	1	0
$[B\,C\,D]$	1	1			

Table 2: Metrics of every ngram

The two metrics indicate the goodness of a possible intermediate symbol w: $num(w)$ indicates how many ICs labeled by w are likely to be generated in parsing; while $ctr(w)$ represents how much w can *contribute* to the generation of CCs. If $ctr(w)$ is larger, the corresponding ICs are more likely to be shared. If ctr is zero, those ICs are surely failed. Therefore the smaller $num(w)$ is and the larger $ctr(w)$ is, the better w would be.

Combining num and ctr, we define a *utility* function for each ngram w in the original grammar:

$$utility(w) = f(num(w), ctr(w)) \qquad (3)$$

where f is a ranking function, satisfying that $f(x, y)$ is larger when x is smaller and y is larger. We will discuss more details about it in Section 5.3.

Using *utility* as the ranking function, we sort all pairs of symbols and choose the best to combine. The formal algorithm is as follows:

S1 For every symbol pair of $\langle v_1, v_2\rangle$ (where v_1 and v_2 can be original symbols or intermediate symbols generated in previous rounds), let w_1 and w_2 be the ngrams of original symbols represented by v_1 and v_2, respectively. Let $w = w_1 w_2$ be the ngram represented by the symbol pair. Compute $utility(w)$.

S2 Select the ngram w with the highest $utility(w)$, let it be w^* (in case of a tie, select the one with a smaller num). Let the corresponding symbol pair be $\langle v_1^*, v_2^*\rangle$.

S3 Add a new intermediate symbol v^*, and replace all the occurrences of $\langle v_1^*, v_2^*\rangle$ on the right hand sides of rules with v^*.

S4 Add a new rule $v^* \rightarrow v_1^* v_2^* : 1.0$.

S5 Repeat S1 $\sim$ S4, until there are no rules with more than two symbols on the right hand side.

5.2 Metrics computing

In this section, we discuss how to compute num and ctr in details.

Computing ctr is straightforward. First we get final parses like in Figure 2 for training sentences. From a final parse, we traverse along every parent node and enumerate every subsequence of its child nodes. For example in Figure 2, from the parent node of $X : [0, 4]$, we can enumerate the following: $[A\,B] : [0, 2]$, $[A\,B\,C] : [0, 3]$, $[A\,B\,C\,D] : [0, 4]$, $[B\,C]$:[1, 3], $[B\,C\,D]$:[1, 4], $[C\,D]$:[2, 4]. We add 1 to all the ctr of these ngrams, respectively.

To compute num, we resort to the same idea of dynamic programming as in CKY. We perform a normal left binarization except that we add all ngrams in the original grammar G as intermediate symbols into the binarized grammar G'. For example, for the rule of $S \rightarrow A\,B\,C : p$, the constructed grammar is as follows:

$$\begin{aligned} [A\,B] &\rightarrow A\,B : 1.0 \\ S &\rightarrow [A\,B]\,C : p \\ [B\,C] &\rightarrow B\,C : 1.0 \end{aligned}$$

Using the constructed G', we employ a normal CKY parsing on the training corpus and compute how many constituents are produced for each ngram. The result is num. Suppose the length of the training sentence is n, the original grammar G has N symbols, and the maximum length of rules is k, then the complexity of this method can be written as $O(N^k n^3)$.

5.3 Ranking function

We discuss the details of the ranking function f used to compute the $utility$ of each ngram w. We come up with two forms for f: linear and log-linear

1. linear: $f(x, y) = -\lambda_1 x + \lambda_2 y$
2. log-linear[3]: $f(x, y) = -\lambda_1 \log(x) + \lambda_2 \log(y)$

where λ_1 and λ_2 are non-negative weights subject to $\lambda_1 + \lambda_2 = 1$[4].

We will use development set to determine which form is better and to learn the best weight settings.

6 Combination with other techniques

Binarization usually plays a role of preprocessing in the procedure of parsing. Grammars are binarized before they are fed into the stage of parsing. There are many known works on speeding up the CKY parsing. So we can expect that if we replace the part of binarization by a better one while keeping the subsequent parsing unchanged, the parsing will be more efficient. We will conduct experiment to confirm this idea in the next section.

We would like to make more discussions before we advance to the experiments. The first is about parsing accuracy in combining binarization with other parsing speed-up techniques. Binarization itself does not affect parsing accuracy. When combined with exact inference algorithms, like the iterative CKY (Tsuruoka and Tsujii, 2004), the accuracy will be the same. However, if combined with other inexact pruning techniques like beam-pruning (Goodman, 1997) or coarse-to-fine parsing (Charniak et al., 2006), binarization may interact with those pruning methods in a complicated way to affect parsing accuracy. This is due to different binarizations generate different sets of intermediate symbols. With the same complete constituents, one binarization might derive incomplete constitutes that could be pruned while another binarization may not. This would affect the accuracy. We do not address this interaction on in this paper, but leave it to the future work. In Section 7.3 we will use the iterative CKY for testing.

In addition, we believe there exist some speed-up techniques which are incompatible with our binarization. One such example may be the top-down left-corner filtering (Graham et al., 1980; Moore, 2000), which seems to be only applicable to the process of left binarization. A detailed investigation on this problem will be left to the future work.

The last issue is how our binarization performs on a lexicalized parser, like Collins (1997). Our intuition is that we cannot apply our binarization to Collins (1997). The key fact in lexicalized parsers is that we cannot explicitly write down all rules and compute their probabilities precisely, due to the great number of rules and the severe data sparsity problem. Therefore in Collins (1997) grammar rules are already factorized into a set of probabilities. In order to capture the dependency relationship between lexcial heads Collins (1997) breaks down the rules from head outwards, which prevents us from factorizing them in other ways. Therefore our binarization cannot apply to the lexicalized parser. However, there are state-of-the-art unlexicalized parsers (Klein and Manning, 2003b; Petrov et al., 2006), to which we believe our binarization can be applied.

7 Experiments

We conducted two experiments on Penn Treebank II corpus (Marcus et al., 1994). The first is to compare the effects of different binarizations on parsing and the second is to test the feasibility to combine our work with iterative CKY parsing (Tsuruoka and Tsujii, 2004) to achieve even better efficiency.

7.1 Experimental setup

Following conventions, we learnt the grammar from Wall Street Journal (WSJ) section 2 to 21 and modified it by discarding all functional tags and empty nodes. The parser obtained this way is a pure unlexicalized context-free parser with the raw treebank grammar. Its accuracy turns out to be 72.46% in

[3]For log-linear form, if $num(w) = 0$ (and consequently $ctr(w) = 0$), we set $f(num(w), ctr(w)) = 0$; if $num(w) > 0$ but $ctr(w) = 0$, we set $f(num(w), ctr(w)) = -\infty$.

[4]Since f is used for ranking, the magnitude is not important.

terms of F1 measure, quite the same as 72.62% as stated in Klein and Manning (2003b). We adopt this parser in our experiment not only because of simplicity but also because we focus on parsing efficiency.

For all sentences with no more than 40 words in section 22, we use the first 10% as the development set, and the last 90% as the test set. There are 158 and 1,420 sentences in development set and test set, respectively. We use the whole 2,416 sentences in section 23 as the training set.

We use the development set to determine the better form of the ranking function f as well as to tune its weights. Both metrics of num and ctr are normalized before use. Since there is only one free variable in λ_1 and λ_2, we can just enumerate $0 \leq \lambda_1 \leq 1$, and set $\lambda_2 = 1 - \lambda_1$. The increasing step is firstly set to 0.05 for the approximate location of the optimal weight, then set to 0.001 to learn more precisely around the optimal.

We find that the optimal is 5,773,088 (constituents produced in parsing development set) with $\lambda_1 = 0.014$ for linear form, while for log-linear form the optimal is 5,905,292 with $\lambda_1 = 0.691$. Therefore we determine that the better form for the ranking function is linear with $\lambda_1 = 0.014$ and $\lambda_2 = 0.986$.

The size of each binarized grammar used in the experiment is shown in Table 3. "Original" refers to the raw treebank grammar. "Ours" refers to the learnt binarized grammar by our approach. For the rest please refer to Section 2.

	# of Symbols	# of Rules
Original	72	14,971
Right	10,654	25,553
Left	12,944	27,843
Head	11,798	26,697
Compact	3,644	18,543
Ours	8,407	23,306

Table 3: Grammar size of different binarizations

We also tested whether the size of the training set would have significant effect. We use the first 10%, 20%, $\cdots$, up to 100% of section 23 as the training set, respectively, and parse the development set. We find that all sizes examined have a similar impact, since the numbers of constituents produced are all around 5,780,000. It means the training corpus does not have to be very large.

The entire experiments are conducted on a server with an Intel Xeon 2.33 GHz processor and 8 GB memory.

7.2 Experiment 1: compare among binarizations

In this part, we use CKY to parse the entire test set and evaluate the efficiency of different binarizations.

The for-statement implementation of the inner most loop of CKY will affect the parsing time though it won't affect the number of constituents produced as discussed in Section 3.2. The best implementations may be different for different binarized grammars. We examine M1$\sim$M4, testing their parsing time on the development set. Results show that for right binarization the best method is M3, while for the rest the best is M2. We use the best method for each binarized grammar when comparing the parsing time in Experiment 1.

Table 4 reports the total number of constituents and total time required for parsing the entire test set. It shows that different binarizations have great impacts on the efficiency of CKY. With our binarization, the number of constituents produced is nearly 20% of that required by right binarization and nearly 25% of that by the widely-used left binarization. As for the parsing time, CKY with our binarization is about 2.5 times as fast as with right binarization and about 1.75 times as fast as with left binarization. This illustrates that our binarization can significantly improve the efficiency of the CKY parsing.

Binarization	Constituents	Time (s)
Right	241,924,229	5,747
Left	193,238,759	3,474
Head	166,425,179	3,837
Compact	94,257,478	2,302
Ours	**52,206,466**	**2,182**

Table 4: Performance on test set

Figure 3 reports the detailed number of complete constituents, successful incomplete constituents and failed incomplete constituents produced in parsing. The result proves that our binarization can significantly reduce the number of failed incomplete constituents, by a factor of 10 in contrast with left binarization. Meanwhile, the number of successful incomplete constituents is also reduced by a factor of 2 compared to left binarization.

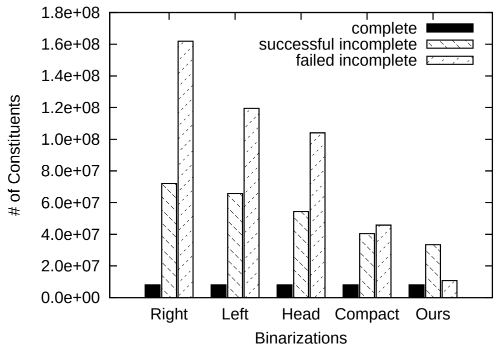


Figure 3: Comparison on various constituents

Another interesting observation is that parsing with a smaller grammar does not always yield a higher efficiency. Our binarized grammar is more than twice the size of compact binarization, but ours is more efficient. It proves that parsing efficiency is related to both the size of grammar in use as well as the number of constituents produced.

In Section 1, we used an example of "*get the bag and go*" to illustrate that for rules like $X \rightarrow Y\,CC\,Y$, right binarization is more suitable. We also investigated the corresponding linguistic nature that the word to the right of "*and*" is more likely to indicate the true relationship represented by "*and*". We argued that a better binarization can reflect such linguistic nature of the input language. To our surprise, our learnt binarization indeed captures this linguistic insight, by binarizing $NP \rightarrow NP\,CC\,NP$ from right to left.

Finally, we would like to acknowledge the limitation of our assumption made in Section 3.2. Table 4 shows that the parsing time of CKY is not always monotonic increasing with the number of constituents produced. Head binarization produces fewer constituents than left binarization but consumes more parsing time.

7.3 Experiment 2: combine with iterative CKY

In this part, we test the performance of combining our binarization with the iterative CKY (Tsuruoka and Tsujii, 2004) (henceforth T&T) algorithm.

Iterative CKY is a procedure of multiple passes of normal CKY: in each pass, it uses a threshold to prune bad constituents; if it cannot find a successful parse in one pass, it will relax the threshold and start another; this procedure is repeated until a successful parse is returned. T&T used left binarization. We re-implement their experiments and combine iterative CKY with our binarization. Note that iterative CKY is an exact inference algorithm that guarantees to return the optimal parse. As discussed in Section 6, the parsing accuracy is not changed in this experiment.

T&T used a held-out set to learn the best step of threshold decrease. They reported that the best step was 11 (in log-probability). We found that the best step was indeed 11 for left binarization; for our binarizaiton, the best step was 17. T&T used M4 as the for-statement implementation of CKY. In this part, we follow the same method.

The result is shown in Table 5. We can see that iterative CKY can achieve better performance by using a better binarization. We also see that the reduction by binarization with pruning is less significant than without pruning. It seems that the pruning itself in iterative CKY can counteract the reduction effect of binarization to some extent. Still the best performance is archieved by combining iterative CKY with a better binarization.

CKY + Binarization	Constituents	Time (s)
Tsuruoka and Tsujii (2004)		
CKY + Left	45,406,084	1,164
Iterative CKY + Left	17,520,427	613
Reimplement		
CKY + Left	52,128,941	932
CKY + Ours	14,892,203	571
Iterative CKY + Left	23,267,594	377
Iterative CKY + Ours	**10,966,272**	**314**

Table 5: Combining with iterative CKY parsing

8 Related work

Almost all work on parsing starts from a binarized grammar. Usually binarization plays a role of preprocessing. Left binarization is widely used (Aho and Ullman, 1972; Charniak et al., 1998; Tsuruoka and Tsujii, 2004) while right binarization is rarely used in the literature. Compact binarization was introduced in Schmid (2004), based on the intuition that a more compact grammar will help acheive a highly efficient CKY parser, though from our experiment it is not always true.

We define the fashion of binarizations in Section 2, where we encode an intermediate symbol using the ngrams of original symbols (content) it derives. This encoding is known as the Inside-Trie (I-Trie) in Klein and Manning (2003a), in which they also mentioned another encoding called Outside-Trie (O-Trie). O-Trie encodes an intermediate symbol using the its parent and the symbols surrounding it in the original rule (context). Klein and Manning (2003a) claimed that O-Trie is superior for calculating estimates for A* parsing. We plan to investigate binarization defined by O-Trie in the future.

Both I-Trie and O-Trie are equivalent encodings, resulting in equivalent grammars, because they both encode using the complete content or context information of an intermediate symbol. If we use part of the information to encode, for example just parent in O-Trie case, the encoding will be non-equivalent.

Proper non-equivalent encodings are used to generalize the grammar and prevent the binarized grammar becoming too specific (Charniak et al., 2006). It is equipped with head binarization to help improve parsing accuracy, following the traditional linguistic insight that phrases are organized around the head (Collins, 1997; Klein and Manning, 2003b). In contrast, we focus our attention on parsing efficiency not accuracy in this paper.

Binarization also attracts attention in the syntax-based models for machine translation, where translation can be modeled as a parsing problem and binarization is essential for efficient parsing (Zhang et al., 2006; Huang, 2007).

Wang et al. (2007) employs binarization to decompose syntax trees to acquire more re-usable translation rules in order to improve translation accuracy. Their binarization is restricted to be a mixture of left and right binarization. This constraint may decrease the power of binarization when applied to speeding up parsing in our problem.

9 Conclusions and future work

We have studied the impact of grammar binarization on parsing efficiency and presented a novel binarization which utilizes rich information learnt from training corpus. Experiments not only showed that our learnt binarization outperforms other existing ones in terms of parsing efficiency, but also demonstrated the feasibility to combine our binarization with known parsing speed-up techniques to achieve even better performance.

An advantage of our approach to finding a good binarization would be that the training corpus does not need to be parsed sentences. Only POS tagged sentences will suffice for training. This will save the effort to adapt the model to a new domain.

Our approach is based on the assumption that the efficiency of CKY parsing is primarily determined by the number of constituents produced. This is a fairly sound one, but not always true, as shown in Section 7.2. One future work will be relaxing the assumption and finding a better appraoch.

Another future work will be to apply our work to chart parsing. It is known that binarization is also essential for an $O(n^3)$ complexity of chart parsing, where *dotted rules* are used to binarize the grammar implicitly from left. As shown in Charniak et al. (1998), we can binarize explicitly and use intermediate symbols to replace dotted rules in chart parsing. Therefore chart parsing can use multiple binarizations. We expect that a better binarization will also help improve the efficiency of chart parsing.

Acknowledgements

We thank the anonymous reviwers for their pertinent comments, Yoshimasa Tsuruoka for the detailed explanations on his referred paper, Yunbo Cao, Shujian Huang, Zhenxing Wang , John Blitzer and Liang Huang for their valuable suggestions in preparing the paper.

References

Aho, A. V. and Ullman, J. D. (1972). *The theory of parsing, translation, and compiling*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Charniak, E., Goldwater, S., and Johnson, M. (1998). Edge-based best-first chart parsing. In *Proceedings of the Six Workshop on Very Large Corpora*, pages 127–133.

Charniak, E., Johnson, M., Elsner, M., Austerweil, J., Ellis, D., Haxton, I., Hill, C., Shrivaths, R., Moore, J., Pozar, M., and Vu, T. (2006). Multi-level coarse-to-fine pcfg parsing. In *HLT-NAACL*.

Collins, M. (1997). Three generative, lexicalised models for statistical parsing. In *ACL*.

Earley, J. (1970). An efficient context-free parsing algorithm. *Commun. ACM*, 13(2):94–102.

Goodman, J. (1997). Global thresholding and multiple-pass parsing. In *EMNLP*.

Graham, S. L., Harrison, M. A., and Ruzzo, W. L. (1980). An improved context-free recognizer. *ACM Trans. Program. Lang. Syst.*, 2(3):415–462.

Huang, L. (2007). Binarization, synchronous binarization, and target-side binarization. In *Proceedings of SSST, NAACL-HLT 2007 / AMTA Workshop on Syntax and Structure in Statistical Translation*, pages 33–40, Rochester, New York. Association for Computational Linguistics.

Kasami, T. (1965). An efficient recognition and syntax analysis algorithm for context-free languages. Technical Report AFCRL-65-758, Air Force Cambridge Research Laboratory, Bedford, Massachusetts.

Kay, M. (1980). Algorithm schemata and data structures in syntactic processing. Technical Report CSL80-12, Xerox PARC, Palo Alto, CA.

Klein, D. and Manning, C. D. (2001). Parsing and hypergraphs. In *IWPT*.

Klein, D. and Manning, C. D. (2003a). A* parsing: Fast exact viterbi parse selection. In *HLT-NAACL*.

Klein, D. and Manning, C. D. (2003b). Accurate unlexicalized parsing. In *ACL*.

Marcus, M. P., Kim, G., Marcinkiewicz, M. A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., and Schasberger, B. (1994). The penn treebank: Annotating predicate argument structure. In *HLT-NAACL*.

Moore, R. C. (2000). Improved left-corner chart parsing for large context-free grammars. In *IWPT*.

Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *ACL*.

Schmid, H. (2004). Efficient parsing of highly ambiguous context-free grammars with bit vectors. In *COLING*.

Tsuruoka, Y. and Tsujii, J. (2004). Iterative cky parsing for probabilistic context-free grammars. In *IJCNLP*.

Wang, W., Knight, K., and Marcu, D. (2007). Binarizing syntax trees to improve syntax-based machine translation accuracy. In *EMNLP-CoNLL*.

Younger, D. H. (1967). Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208.

Zhang, H., Huang, L., Gildea, D., and Knight, K. (2006). Synchronous binarization for machine translation. In *HLT-NAACL*.

Sentence Fusion via Dependency Graph Compression

Katja Filippova and **Michael Strube**
EML Research gGmbH
Schloss-Wolfsbrunnenweg 33
69118 Heidelberg, Germany
http://www.eml-research.de/nlp

Abstract

We present a novel unsupervised sentence fusion method which we apply to a corpus of biographies in German. Given a group of related sentences, we align their dependency trees and build a dependency graph. Using integer linear programming we compress this graph to a new tree, which we then linearize. We use GermaNet and Wikipedia for checking semantic compatibility of co-arguments. In an evaluation with human judges our method outperforms the fusion approach of Barzilay & McKeown (2005) with respect to readability.

1 Introduction

Automatic text summarization is a rapidly developing field in computational linguistics. Summarization systems can be classified as either extractive or abstractive ones (Spärck Jones, 1999). To date, most systems are extractive: sentences are selected from one or several documents and then ordered. This method exhibits problems, because input sentences very often overlap and complement each other at the same time. As a result there is a trade-off between *non-redundancy* and *completeness* of the output. Although the need for abstractive approaches has been recognized before (e.g. McKeown et al. (1999)), so far almost all attempts to get closer to abstractive summarization using scalable, statistical techniques have been limited to sentence compression.

The main reason why there is little progress on abstractive summarization is that this task seems to require a conceptual representation of the text which is not yet available (see e.g. Hovy (2003, p.589)). Sentence fusion (Barzilay & McKeown, 2005), where a new sentence is generated from a group of related sentences and where complete semantic and conceptual representation is not required, can be seen as a middle-ground between extractive and abstractive summarization. Our work regards a corpus of biographies in German where multiple documents about the same person should be merged into a single one. An example of a fused sentence (3) with the source sentences (1,2) is given below:

(1)

Bohr	studierte	an	der	Universität	Kopenhagen
Bohr	studied	at	the	University	Copenhagen
und	erlangte	dort	seine	Doktorwürde.	
and	got	there	his	PhD	

'Bohr studied at the University of Copenhagen and got his PhD there'

(2)

Nach	dem	Abitur	studierte	er	Physik	und
After	the	school	studied	he	physics	and
Mathematik	an	der	Universität	Kopenhagen.		
mathematics	at	the	University	Copenhagen		

'After school he studied physics and mathematics at the University of Copenhagen'

(3)

Nach	dem	Abitur	studierte	Bohr	Physik	und
After	the	school	studied	Bohr	physics	and
Mathematik	an	der	Universität	Kopenhagen		
mathematics	at	the	University	Copenhagen		
und	erlangte	dort	seine	Doktorwürde.		
and	got	there	his	PhD		

'After school Bohr studied physics and mathematics at the University of Copenhagen and got his PhD there'

Having both (1) and (2) in a summary would make it redundant. Selecting only one of them would not give all the information from the input. (3), fused from both (1) and (2), conveys the necessary information without being redundant and is more appropriate for a summary.

To this end, we present a novel sentence fusion method based on dependency structure alignment and semantically and syntactically informed phrase aggregation and pruning. We address the problem in an unsupervised manner and use integer linear programming (ILP) to find a globally optimal solution. We argue that our method has three important advantages compared to existing methods. First, we address the grammaticality issue empirically by means of knowledge obtained from an automatically parsed corpus. We do not require such resources as subcategorization lexicons or hand-crafted rules, but decide to retain a dependency based on its syntactic importance score. The second point concerns integrating semantics. Being definitely important, *"this source of information remains relatively unused in work on aggregation*[1] *within NLG"* (Reiter & Dale, 2000, p.141). To our knowledge, in the text-to-text generation field, we are the first to use semantic information not only for alignment but also for aggregation in that we check coarguments' compatibility. Apart from that, our method is not limited to sentence fusion and can be easily applied to sentence compression. In Filippova & Strube (2008) we compress English sentences with the same approach and achieve state-of-the-art performance.

The paper is organized as follows: Section 2 gives an overview of related work and Section 3 presents our data. Section 4 introduces our method and Section 5 describes the experiments and discusses the results of the evaluation. The conclusions follow in the final section.

2 Related Work

Most studies on text-to-text generation concern sentence compression where the input consists of exactly one sentence (Jing, 2001; Hori & Furui, 2004; Clarke & Lapata, 2008, inter alia). In such setting, redundancy, incompleteness and compatibility issues do not arise. Apart from that, there is no obvious way of how existing sentence compression methods can be adapted to sentence fusion.

Barzilay & McKeown (2005) present a sentence fusion method for multi-document news summarization which crucially relies on the assumption that information appearing in many sources is important. Consequently, their method produces an intersection of input sentences by, first, finding the centroid of the input, second, augmenting it with information from other sentences and, finally, pruning a predefined set of constituents (e.g. PPs). The resulting structure is not necessarily a tree and allows for extraction of several trees, each of which can be linearized in many ways.

Marsi & Krahmer (2005) extend the approach of Barzilay & McKeown to do not only *intersection* but also *union* fusion. Like Barzilay & McKeown (2005), they find the best linearization with a language model which, as they point out, often produces inadequate rankings being unable to deal with word order, agreement and subcategorization constraints. In our work we aim at producing a valid dependency tree structure so that most grammaticality issues are resolved *before* the linearization stage.

Wan et al. (2007) introduce a global revision method of how a novel sentence can be generated from a set of input words. They formulate the problem as a search for a maximum spanning tree which is incrementally constructed by connecting words or phrases with dependency relations. The grammaticality issue is addressed by a number of hard constraints. As Wan et al. point out, one of the problems with their method is that the output built up from dependencies found in a corpus might have a meaning different from the intended one. Since we build our trees from the input dependencies, this problem does not arise with our method. Apart from that, in our opinion, the optimization formulation we adopt is more appropriate as it allows to integrate many constraints without complex rescoring rules.

3 Data

The comparable corpus we work with is a collection of about 400 biographies in German gathered from

[1]We follow Barzilay & McKeown (2005) and refer to aggregation within text-to-text generation as sentence fusion.

the Internet[2]. These biographies describe 140 different people, and the number of articles for one person ranges from 2 to 4, being 3 on average. Despite obvious similarities between articles about one person, neither identical content nor identical ordering of information can be expected.

Fully automatic preprocessing in our system comprises the following steps: sentence boundaries are identified with a Perl CPAN module[3]. Then the sentences are split into tokens and the TnT tagger (Brants, 2000) and the TreeTagger (Schmid, 1997) are used for tagging and lemmatization respectively. Finally, the biographies are parsed with the CDG dependency parser (Foth & Menzel, 2006). We also identify references to the biographee (pronominal as well as proper names) and temporal expressions (absolute and relative) with a few rules.

4 Our Method

Groups of related sentences serve as input to a sentence fusion system and thus need to be identified first (4.1). Then the dependency trees of the sentences are modified (4.2) and aligned (4.3). Syntactic importance (4.4) and word informativeness (4.5) scores are used to extract a new dependency tree from a graph of aligned trees (4.6). Finally, the tree is linearized (4.7).

4.1 Sentence Alignment

Sentence alignment for comparable corpora requires methods different from those used in machine translation for parallel corpora. For example, given two biographies of a person, one of them may follow the timeline from birth to death whereas the other may group events thematically or tell only about the scientific contribution of the person. Thus one cannot assume that the sentence order or the content is the same in two biographies. Shallow methods like word or bigram overlap, (weighted) cosine or Jaccard similarity are appealing as they are cheap and robust. In particular, Nelken & Schieber (2006) demonstrate the efficacy of a sentence-based *tf*idf* score when applied to comparable corpora. Following them, we define the similarity of two sentences *sim*(s_1, s_2) as

$$\frac{S_1 \cdot S_2}{|S_1| \cdot |S_2|} = \frac{\sum_t w_{S_1}(t) \cdot w_{S_2}(t)}{\sqrt{\sum_t w^2_{S_1}(t) \sum_t w^2_{S_2}(t)}} \qquad (1)$$

where S is the set of all lemmas but stop-words from s, and $w_S(t)$ is the weight of the term t:

$$w_S(t) = S(t)\frac{1}{N_t} \qquad (2)$$

where $S(t)$ is the indicator function of S, N_t is the number of sentences in the biographies of one person which contain t. We enhance the similarity measure by looking up synonymy in GermaNet (Lemnitzer & Kunze, 2002).

We discard identical or nearly identical sentences ($sim(s_1, s_2) > 0.8$) and greedily build sentence clusters using a hierarchical groupwise-average technique. As a result, one sentence may belong to one cluster at most. These sentence clusters serve as input to the fusion algorithm.

4.2 Dependency Tree Modification

We apply a set of transformations to a dependency tree to emphasize its important properties and eliminate unimportant ones. These transformations are necessary for the compression stage. An example of a dependency tree and its modifed version are given in Fig. 1.

PREP preposition nodes (*an, in*) are removed and placed as labels on the edges to the respective nouns;

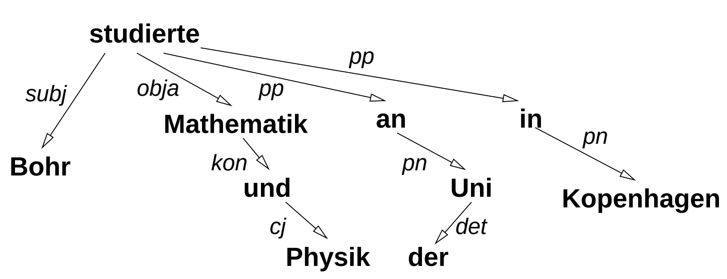
CONJ a chain of conjuncts (*Mathematik und Physik*) is split and each node is attached to the parent node (*studierte*) provided they are not verbs;

APP a chain of words analyzed as appositions by CDG (*Niels Bohr*) is collapsed into one node;

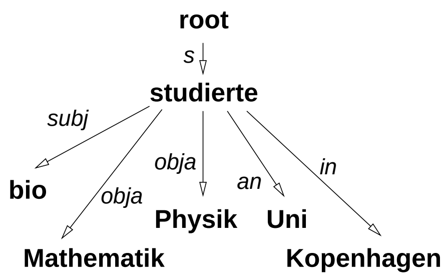
FUNC function words like determiners (*der*), auxiliary verbs or negative particles are removed from the tree and memorized with their lexical heads (memorizing negative particles preserves negation in the output);

[2]http://de.wikipedia.org, http://home.datacomm.ch/biografien, http://biographie.net/de, http://www.weltchronik.de/ws/bio/main.htm, http://www.brockhaus-suche.de/suche

[3]http://search.cpan.org/~holsten/Lingua-DE-Sentence-0.07/Sentence.pm



(a) Dependency tree



(b) Modified tree

Figure 1: The dependency tree of the sentence *Bohr studierte Mathematik und Physik an der Uni in Kopenhagen (Bohr studied mathematics and physics at university in Copenhagen)* as produced by the parser (a) and after all transformations applied (b)

ROOT every dependency tree gets an explicit root which is connected to every verb node;

BIO all occurrences of the biographee (*Niels Bohr*) are replaced with the *bio* tag.

4.3 Node Alignment

Once we have a group of two to four strongly related sentences and their transformed dependency trees, we aim at finding the best node alignment. We use a simple, fast and transparent method and align any two words provided that they

1. are content words;
2. have the same part-of-speech;
3. have identical lemmas or are synonyms.

In case of multiple possibilities, which are extremely rare in our data, the choice is made randomly. By merging all aligned nodes we get a dependency graph which consists of all dependencies from the input trees. In case it contains a cycle, one of the alignments from the cycle is eliminated.

We prefer this very simple method to bottom-up ones (Barzilay & McKeown, 2005; Marsi & Krahmer, 2005) for two main reasons. Pursuing local subtree alignments, bottom-up methods may leave identical words unaligned and thus prohibit fusion of complementary information. On the other hand, they may force alignment of two unrelated words if the subtrees they root are largely aligned. Although in some cases it helps discover paraphrases, it considerably increases chances of generating ungrammatical output which we want to avoid at any cost.

4.4 Syntactic Importance Score

Given a dependency graph we want to get a new dependency tree from it. Intuitively, we want to retain obligatory dependencies (e.g. *subject*) while removing less important ones (e.g. *adv*). When deciding on pruning an argument, previous approaches either used a set of hand-crafted rules (e.g. Barzilay & McKeown (2005)), or utilized a subcategorization lexicon (e.g. Jing (2001)). The hand-crafted rules are often too general to ensure a grammatical argument structure for different verbs (e.g. *PPs can be pruned*). Subcategorization lexicons are not readily available for many languages and cover only verbs. E.g. they do not tell that the noun *son* is very often modified by a PP using the preposition *of*, as in *the son of Niels Bohr*, and that the NP without a PP modifier may appear incomplete.

To overcome these problems, we decide on pruning an edge by estimating the conditional probability of its label given its head, $P(l|h)$[4]. For example, $P(subj|studieren)$ – the probability of the label *subject* given the verb *study* – is higher than $P(in|studieren)$, and therefore the subject will be preserved whereas the prepositional label and thus the whole PP can be pruned, if needed. Table 1 presents the probabilities of several labels given that the head is *studieren* and shows that some prepositions are more important than other ones. Note that if we did not apply the PREP modification we would be unable to distinguish between different prepositions and could only calculate $P(pp|studieren)$

[4]The probabilities are calculated from a corpus of approx. 3,000 biographies from Wikipedia which we annotated automatically as described in Section 3.

which would not be very informative.

subj	obja	in	an	nach	mit	zu
0.88	0.74	0.44	0.42	0.09	0.02	0.01

Table 1: Probabilities of *subj, obja(ccusative), in, at, after, with, to* given the verb *studieren* (*study*)

4.5 Word Informativeness Score

We also want to retain informative words in the output tree. There are many ways in which word importance can be defined. Here, we use a formula introduced by Clarke & Lapata (2008) which is a modification of the significance score of Hori & Furui (2004):

$$I(w_i) = \frac{l}{N} \cdot f_i \log \frac{F_A}{F_i} \tag{3}$$

w_i is the topic word (either noun or verb), f_i is the frequency of w_i in the aligned biographies, F_i is the frequency of w_i in the corpus, and F_A is the sum of frequencies of all topic words in the corpus. l is the number of clause nodes above w and N is the maximum level of embedding of the sentence which w belongs to. By defining word importance differently, e.g. as relatedness of a word to the topic, we could apply our method to topic-based summarization (Krahmer et al., 2008).

4.6 New Sentence Generation

We formulate the task of getting a tree from a dependency graph as an optimization problem and solve it with ILP[5]. In order to decide which edges of the graph to remove, for each directed dependency edge from head h to word w we introduce a binary variable $x^l_{h,w}$, where *l* stands for the label of the edge:

$$x^l_{h,w} = \begin{cases} 1 & \text{if the dependency is preserved} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The goal is to find a subtree of the graph which gets the highest score of the objective function (5) to which both the probability of dependencies ($P(l|h)$) and the importance of dependent words ($I(w)$) contribute:

[5]We use `lp_solve` in our implementation `http://sourceforge.net/projects/lpsolve`.

$$f(X) = \sum_x x^l_{h,w} \cdot P(l|h) \cdot I(w) \tag{5}$$

The objective function is subject to four types of constraints presented below (W stands for the set of graph nodes minus root, i.e. the set of words).

STRUCTURAL constraints allow to get a tree from the graph: (6) ensures that each word has one head at most. (7) ensures connectivity in the tree. (8) is optional and restricts the size of the resulting tree to α words ($\alpha = \min(0.\bar{6} \cdot |W|, 10)$).

$$\forall w \in W, \sum_{h,l} x^l_{h,w} \leq 1 \tag{6}$$

$$\forall w \in W, \sum_{h,l} x^l_{h,w} - \frac{1}{|W|} \sum_{u,l} x^l_{w,u} \geq 0 \tag{7}$$

$$\sum_x x^l_{h,w} \leq \alpha \tag{8}$$

SYNTACTIC constraints ensure the syntactic validity of the output tree and explicitly state which arguments should be preserved. We have only one syntactic constraint which guarantees that a subordinating conjunction (*sc*) is preserved (9) if and only if the clause it belongs to serves as a subordinate clause (*sub*) in the output.

$$\forall x^{sc}_{w,u}, \sum_{h,l} x^{sub}_{h,w} - x^{sc}_{w,u} = 0 \tag{9}$$

SEMANTIC constraints restrict coordination to semantically compatible elements. The idea behind these constraints is the following (see Fig. 2). It can be that one sentence says *He studied math* and another one *He studied physics*, so the output may unite the two words under coordination: *He studied math and physics*. But if the input sentences are *He studied physics* and *He studied sciences*, then one should not unite both, because *sciences* is the generalization of *physics*. Neither should one unite two unrelated words: *He studied with pleasure* and *He studied with Bohr* cannot be fused into *He studied with pleasure and Bohr*.

To formalize these intuitions we define two functions *hm(w,u)* and *rel(w,u)*: *hm(w,u)* is a binary function, whereas *rel(w,u)* returns a value from $[0, 1]$. We

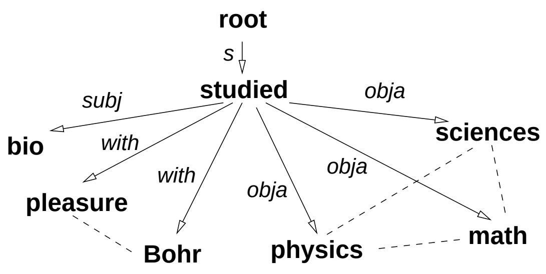


Figure 2: Graph obtained from sentences *He studied sciences with pleasure* and *He studied math and physics with Bohr*

also introduce additional variables $y^l_{w,u}$ (represented by dashed lines in Fig. 2):

$$y^l_{w,u} = \begin{cases} 1 & \text{if } \exists h, l : x^l_{h,w} = 1 \wedge x^l_{h,u} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

For two edges sharing a head and having identical labels to be retained we check in GermaNet and in the taxonomy derived from Wikipedia (Kassner et al., 2008) that their dependents are not in the *h*yponymy or *m*eronymy relation (11). We prohibit verb coordination unless it is found in one of the input sentences. If the dependents are nouns, we also check that their semantic relatedness as measured with WikiRelate! (Strube & Ponzetto, 2006) is above a certain threshold (12). We empirically determined the value of $\beta = 0.36$ by calculating an average similarity of coordinated nouns in the corpus.

$$\forall y^l_{w,u}, \mathtt{hm}(w,u) \cdot y^l_{w,u} = 0 \tag{11}$$

$$\forall y^l_{w,u}, (\mathtt{rel}(w,u) - \beta) \cdot y^l_{w,u} \geq 0 \tag{12}$$

(11) prohibits that *physics* (or *math*) and *sciences* appear together since, according to GermaNet, *physics* (*Physik*) is a hyponym of *science* (*Wissenschaft*). (12) blocks taking both *pleasure* (*Freude*) and *Bohr* because *rel(Freude,Bohr)* $= 0.17$. *math* and *physics* are neither in *ISA*, nor *part-of* relation and are sufficiently related (*rel(Mathematik, Physik)* $= 0.67$) to become conjuncts.

META constraints (equations (13) and (14)) guarantee that $y^l_{w,u} = x^l_{h,w} \times x^l_{h,u}$ i.e. they ensure that the semantic constraints are applied only if both the labels from h to w and from h to u are preserved.

$$\forall y^l_{w,u}, x^l_{h,w} + x^l_{h,u} \geq 2y^l_{w,u} \tag{13}$$

$$\forall y^l_{w,u}, 1 - x^l_{h,w} + 1 - x^l_{h,u} \geq 1 - y^l_{w,u} \tag{14}$$

4.7 Linearization

The "overgenerate-and-rank" approach to statistical surface realization is very common (Langkilde & Knight, 1998). Unfortunately, in its simplest and most popular version, it ignores syntactical constraints and may produce ungrammatical output. For example, an inviolable rule of German grammar states that the finite verb must be in the second position in the main clause. Since it is hard to enforce such rules with an ngram language model, syntax-informed linearization methods have been developed for German (Ringger et al., 2004; Filippova & Strube, 2007). We apply our recent method to order constituents and, using the CMU toolkit (Clarkson & Rosenfeld, 1997), build a trigram language model from Wikipedia (approx. 1GB plain text) to find the best word order within constituents. Some constraints on word order are inferred from the input. Only interclause punctuation is generated.

5 Experiments and Evaluation

We choose Barzilay & McKeown's system as a non-trivial baseline since, to our knowledge, there is no other system which outperforms theirs (Sec. 5.1). It is important for us to evaluate the fusion part of our system, so the input and the linearization module of our method and the baseline are identical. We are also interested in how many errors are due to the linearization module and thus define the readability upper bound (Sec. 5.2). We further present and discuss the experiments (Sec. 5.3 and 5.5).

5.1 Baseline

The algorithm of Barzilay & McKeown (2005) proceeds as follows: Given a group of related sentences, a dependency tree is built for each sentence. These trees are modified so that grammatical features are eliminated from the representation and memorized; noun phrases are flattened to facilitate alignment. A locally optimal pairwise alignment of modified dependency trees is recursively found with WordNet and a paraphrase lexicon. From the alignment costs the centroid of the group is identified. Then this tree is augmented with information from other trees given that it appears in at least half of the sentences from this group. A rule-based pruning module prunes optional constituents, such as PPs or relative clauses. The linearization of the resulting tree (or graph) is done with a trigram language model.

To adapt this system to German, we use the GermaNet API (Gurevych & Niederlich, 2005) instead of WordNet. We do not use a paraphrase lexicon, because there is no comparable corpus of sufficient size available for German. We readjust the alignment parameters of the system to prevent dissimilar nodes from being aligned. The input to the algorithm is generated as described in Sec. 4.1. The linearization is done as described in Sec. 4.7. In cases when there is a graph to linearize, all possible trees covering the maximum number of nodes are extracted from it and linearized. The most probable string is selected as the final output with a language model. For the rest of the reimplementation we follow the algorithm as presented.

5.2 Readability Upper Bound

To find the upper bound on readability, we select one sentence from the input randomly, parse it and linearize the dependency tree as described in Sec. 4.7. This way we obtain a sentence which may differ in form from the input sentences but whose content is identical to one of them.

5.3 Experiments

It is notoriously difficult to evaluate generation and summarization systems as there are many dimensions in which the quality of the output can be assessed. The goal of our present evaluation is in the first place to check whether our method is able to produce sensible output.

We evaluated the three systems (GRAPH-COMPRESSION, BARZILAY & MCKEOWN and READABILITY UB) with 50 native German speakers on 120 fused sentences generated from 40 randomly drawn related sentences groups (3×40). In an online experiment, the participants were asked to read a fused sentence preceded by the input and to rate its readability (*read*) and informativity in respect to the input (*inf*) on a five point scale. The experiment was designed so that every participant rated 40 sentences in total. No participant saw two sentences generated from the same input. The results are presented in Table 2. *len* is an average length in words of the output.

	read	*inf*	*len*
READABILITY UB	4.0	3.5	12.9
BARZILAY & MCKEOWN	3.1	3.0	15.5
GRAPH-COMPRESSION	3.7	3.1	13.0

Table 2: Average readability and informativity on a five point scale, average length in words

5.4 Error Analysis

The main disadvantage of our method, as well as other methods designed to work on syntactic structures, is that it requires a very accurate parser. In some cases, errors in the preprocessing made extracting a valid dependency tree impossible. The poor rating of READABILITY UB also shows that errors of the parser and of the linearization module affect the output considerably.

Although the semantic constraints ruled out many anomalous combinations, the limited coverage of GermaNet and the taxonomy derived from Wikipedia was the reason for some semantic oddities in the sentences generated by our method. For example, it generated phrases like *aus England und Großbritannien* (*from England and Great Britain*). A larger taxonomy would presumably increase the recall of the semantic constraints which proved helpful. Such errors were not observed in the output of the baseline because it does not fuse within NPs.

Both the baseline and our method made subcategorization errors, although these are more common for the baseline which aligns not only synonyms but also verbs which share some arguments. Also, the baseline pruned some PPs necessary for a sentence to be complete. For example, it pruned *an der Atombombe* (*on the atom bomb*) and generated an incomplete sentence *Er arbeitete* (*He worked*). For the baseline, alignment of flattened NPs instead of words caused generating very wordy and redundant sentences when the input parse trees were incorrect. In other cases, our method made mistakes in linearizing constituents because it had to rely on a language model whereas the baseline used unmodified constituents from the input. Absense of intra-clause commas caused a drop in readability in some otherwise grammatical sentences.

5.5 Discussion

A paired t-test revealed significant differences between the readability ratings of the three systems ($p = 0.01$) but found no significant differences between the informativity scores of our system and the baseline. Some participants reported informativity hard to estimate and to be assessable for grammatical sentences only. The higher readability rating of our method supports our claim that the method based on syntactic importance score and global constraints generates more grammatical sentences than existing systems. An important advantage of our method is that it addresses the subcategorization issue directly without shifting the burden of selecting the right arguments to the linearization module. The dependency structure it outputs is a tree and not a graph as it may happen with the method of Barzilay & McKeown (2005). Moreover, our method can distinguish between more and less obligatory arguments. For example, it knows that *at* is more important than *to* for *study* whereas for *go* it is the other way round. Unlike our differentiated approach, the baseline rule states that PPs can generally be pruned.

Since the baseline generates a new sentence by modifying the tree of an input sentence, in some cases it outputs a compression of this sentence. Unlike this, our method is not based on an input tree and generates a new sentence without being biased to any of the input sentences.

Our method can also be applied to non-trivial sentence compression, whereas the baseline and similar methods, such as Marsi & Krahmer (2005), would then boil down to a few very general pruning rules. We tested our method on the English compression corpus[6] and evaluated the compressions automatically the same way as Clarke & Lapata (2008) did. The results (Filippova & Strube, 2008) were as good as or significantly better than the state-of-the-art, depending on the choice of dependency parser.

[6]The corpus is available from `http://homepages.inf.ed.ac.uk/s0460084/data`.

6 Conclusions

We presented a novel sentence fusion method which formulates the fusion task as an optimization problem. It is unsupervised and finds a globally optimal solution taking semantics, syntax and word informativeness into account. The method does not require hand-crafted rules or lexicons to generate grammatical output but relies on the syntactic importance score calculated from an automatically parsed corpus. An experiment with native speakers demonstrated that our method generates more grammatical sentences than existing systems.

There are several directions to explore in the future. Recently query-based sentence fusion has been shown to be a better defined task than generic sentence fusion (Krahmer et al., 2008). By modifying the word informativeness score, e.g. by giving higher scores to words semantically related to the query, one could force our system to retain words relevant to the query in the output. To generate coherent texts we plan to move beyond sentence generation and add discourse constraints to our system.

Acknowledgements: This work has been funded by the Klaus Tschira Foundation, Heidelberg, Germany. The first author has been supported by a KTF grant (09.009.2004). Part of the data has been used with a permission of Bibliographisches Institut & F. A. Brockhaus AG, Mannheim, Germany. We would like to thank the participants in our online evaluation. We are also grateful to Regina Barzilay and the three reviewers for their helpful comments.

References

Barzilay, Regina & Kathleen R. McKeown (2005). Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–327.

Brants, Thorsten (2000). TnT – A statistical Part-of-Speech tagger. In *Proceedings of the 6th Conference on Applied Natural Language Processing*, Seattle, Wash., 29 April – 4 May 2000, pp. 224–231.

Clarke, James & Mirella Lapata (2008). Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429.

Clarkson, Philip & Ronald Rosenfeld (1997). Statistical language modeling using the CMU-Cambridge toolkit. In *Proceedings of the 5th European Conference on Speech Communication and Technology*, Rhodes, Greece, 22-25 September 1997, pp. 2707–2710.

Filippova, Katja & Michael Strube (2007). Generating constituent order in German clauses. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics,* Prague, Czech Republic, 23–30 June 2007, pp. 320–327.

Filippova, Katja & Michael Strube (2008). Dependency tree based sentence compression. In *Proceedings of the 5th International Conference on Natural Language Generation,* Salt Fork, Ohio, 12–14 June 2008, pp. 25–32.

Foth, Kilian & Wolfgang Menzel (2006). Hybrid parsing: Using probabilistic models as predictors for a symbolic parser. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics,* Sydney, Australia, 17–21 July 2006, pp. 321–327.

Gurevych, Iryna & Hendrik Niederlich (2005). Accessing GermaNet data and computing semantic relatedness. In *Companion Volume to the Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics,* Ann Arbor, Mich., 25–30 June 2005, pp. 5–8.

Hori, Chiori & Sadaoki Furui (2004). Speech summarization: An approach through word extraction and a method for evaluation. *IEEE Transactions on Information and Systems*, E87-D(1):15–25.

Hovy, Eduard (2003). Text summarization. In Ruslan Mitkov (Ed.), *The Oxford Handbook of Computational Linguistics*, pp. 583–598. Oxford, U.K.: Oxford University Press.

Jing, Hongyan (2001). *Cut-and-Paste Text Summarization*, (Ph.D. thesis). Computer Science Department, Columbia University, New York, N.Y.

Kassner, Laura, Vivi Nastase & Michael Strube (2008). Acquiring a taxonomy from the German Wikipedia. In *Proceedings of the 6th International Conference on Language Resources and Evaluation,* Marrakech, Morocco, 26 May – 1 June 2008.

Krahmer, Emiel, Erwin Marsi & Paul van Pelt (2008). Query-based sentence fusion is better defined and leads to more preferred results than generic sentence fusion. In *Companion Volume to the Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics,* Columbus, Ohio, 15–20 June 2008, pp. 193–196.

Langkilde, Irene & Kevin Knight (1998). Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics,* Montréal, Québec, Canada, 10–14 August 1998, pp. 704–710.

Lemnitzer, Lothar & Claudia Kunze (2002). GermaNet – representation, visualization, application. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation,* Las Palmas, Canary Islands, Spain, 29–31 May 2002, pp. 1485–1491.

Marsi, Erwin & Emiel Krahmer (2005). Explorations in sentence fusion. In *Proceedings of the European Workshop on Natural Language Generation,* Aberdeen, Scotland, 8–10 August, 2005, pp. 109–117.

McKeown, Kathleen R., Judith L. Klavans, Vassileios Hatzivassiloglou, Regina Barzilay & Eleazar Eskin (1999). Towards multidocument summarization by reformulation: Progress and prospects. In *Proceedings of the 16th National Conference on Artificial Intelligence,* Orlando, Flo., 18–22 July 1999, pp. 453–460.

Nelken, Rani & Stuart Schieber (2006). Towards robust context-sensitive sentence alignment for monolingual corpora. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics,* Trento, Italy, 3–7 April 2006, pp. 161–168.

Reiter, Ehud & Robert Dale (2000). *Building Natural Language Generation Systems*. Cambridge, U.K.: Cambridge University Press.

Ringger, Eric, Michael Gamon, Robert C. Moore, David Rojas, Martine Smets & Simon Corston-Oliver (2004). Linguistically informed statistical models of constituent structure for ordering in sentence realization. In *Proceedings of the 20th International Conference on Computational Linguistics,* Geneva, Switzerland, 23–27 August 2004, pp. 673–679.

Schmid, Helmut (1997). Probabilistic Part-of-Speech tagging using decision trees. In Daniel Jones & Harold Somers (Eds.), *New Methods in Language Processing*, pp. 154–164. London, U.K.: UCL Press.

Spärck Jones, Karen (1999). Automatic summarizing: Factors and directions. In Inderjeet Mani & Mark T. Maybury (Eds.), *Advances in Automatic Text Summarization*, pp. 1–12. Cambridge, Mass.: MIT Press.

Strube, Michael & Simone Paolo Ponzetto (2006). WikiRelate! Computing semantic relatedness using Wikipedia. In *Proceedings of the 21st National Conference on Artificial Intelligence,* Boston, Mass., 16–20 July 2006, pp. 1419–1424.

Wan, Stephen, Robert Dale, Mark Dras & Cecile Paris (2007). Global revision in summarization: Generating novel sentences with Prim's algorithm. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics,* Melbourne, Australia, 19–21 September, 2007, pp. 226–235.

Revisiting Readability: A Unified Framework for Predicting Text Quality

Emily Pitler
Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA
epitler@seas.upenn.edu

Ani Nenkova
Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA
nenkova@seas.upenn.edu

Abstract

We combine lexical, syntactic, and discourse features to produce a highly predictive model of human readers' judgments of text readability. This is the first study to take into account such a variety of linguistic factors and the first to empirically demonstrate that discourse relations are strongly associated with the perceived quality of text. We show that various surface metrics generally expected to be related to readability are not very good predictors of readability judgments in our Wall Street Journal corpus. We also establish that readability predictors behave differently depending on the task: predicting text readability or ranking the readability. Our experiments indicate that discourse relations are the one class of features that exhibits robustness across these two tasks.

1 Introduction

The quest for a precise definition of text quality—pinpointing the factors that make text flow and easy to read—has a long history and tradition. Way back in 1944 Robert Gunning Associates was set up, offering newspapers, magazines and business firms consultations on clear writing (Gunning, 1952). In education, teaching good writing technique and grading student writing has always been of key importance (Spandel, 2004; Attali and Burstein, 2006). Linguists have also studied various aspects of text flow, with cohesion-building devices in English (Halliday and Hasan, 1976), rhetorical structure theory (Mann and Thompson, 1988) and centering theory (Grosz et al., 1995) among the most influential contributions.

Still, we do not have unified computational models that capture the interplay between various aspects of readability. Most studies focus on a single factor contributing to readability for a given intended audience. The use of rare words or technical terminology for example can make text difficult to read for certain audience types (Collins-Thompson and Callan, 2004; Schwarm and Ostendorf, 2005; Elhadad and Sutaria, 2007). Syntactic complexity is associated with delayed processing time in understanding (Gibson, 1998) and is another factor that can decrease readability. Text organization (discourse structure), topic development (entity coherence) and the form of referring expressions also determine readability. But we know little about the relative importance of each factor and how they combine in determining perceived text quality.

In our work we use texts from the Wall Street Journal intended for an *educated adult audience* to analyze readability factors including vocabulary, syntax, cohesion, entity coherence and discourse. We study the association between these features and reader assigned readability ratings, showing that discourse and vocabulary are the factors most strongly linked to text quality. In the easier task of text quality ranking, entity coherence and syntax features also become significant and the combination of features allows for ranking prediction accuracy of 88%. Our study is novel in the use of gold-standard discourse features for predicting readability and the simultaneous analysis of various readability factors.

2 Related work

2.1 Readability with respect to intended readers

The definition of what one might consider to be a well-written and readable text heavily depends on the intended audience (Schriver, 1989). Obviously, even a superbly written scientific paper will not be perceived as very readable by a lay person and a great novel might not be appreciated by a third grader. As a result, the vast majority of prior work on readability deals with labeling texts with the appropriate school grade level. A key observation in even the oldest work in this area is that the vocabulary used in a text largely determines its readability. More common words are easier, so some metrics measured text readability by the percentage of words that were not among the N most frequent in the language. It was also observed that frequently occurring words are often short, so word length was used to approximate readability more robustly than using a predefined word frequency list. Standard indices were developed based on the link between word frequency/length and readability, such as Flesch-Kincaid (Kincaid, 1975), Automated Readability Index (Kincaid, 1975), Gunning Fog (Gunning, 1952), SMOG (McLaughlin, 1969), and Coleman-Liau (Coleman and Liau, 1975). They use only a few simple factors that are designed to be easy to calculate and are rough approximations to the linguistic factors that determine readability. For example, Flesch-Kincaid uses the average number of syllables per word to approximate vocabulary difficulty and the average number of words per sentence to approximate syntactic difficulty.

In recent work, the idea of linking word frequency and text readability has been explored for making medical information more accessible to the general public. (Elhadad and Sutaria, 2007) classified words in medical texts as familiar or unfamiliar to a general audience based on their frequencies in corpora. When a description of the unfamiliar terms was provided, the perceived readability of the texts almost doubled.

A more general and principled approach to using vocabulary information for readability decisions has been the use of language models. For any given text, it is easy to compute its likelihood under a given language model, i.e. one for text meant for children, or for text meant for adults, or for a given grade level. (Si and Callan, 2001), (Collins-Thompson and Callan, 2004), (Schwarm and Ostendorf, 2005), and (Heilman et al., 2007) used language models to predict the suitability of texts for a given school grade level. But even for this type of task other factors besides vocabulary use are at play in determining readability. Syntactic complexity is an obvious factor: indeed (Heilman et al., 2007) and (Schwarm and Ostendorf, 2005) also used syntactic features, such as parse tree height or the number of passive sentences, to predict reading grade levels. For the task of deciding whether a text is written for an adult or child reader, (Barzilay and Lapata, 2008) found that adding entity coherence to (Schwarm and Ostendorf, 2005)'s list of features improves classification accuracy by 10%.

2.2 Readability as coherence for competent language users

In linguistics and natural language processing, the text properties rather than those of the reader are emphasized. Text coherence is defined as the ease with which a person (tacitly assumed to be a competent language user) understands a text. Coherent text is characterized by various types of cohesive links that facilitate text comprehension (Halliday and Hasan, 1976).

In recent work, considerable attention has been devoted to entity coherence in text quality, especially in relation to information ordering. In many applications such as text generation and summarization, systems need to decide the order in which selected sentences or generated clauses should be presented to the user. Most models attempting to capture local coherence between sentences were based on or inspired by centering theory (Grosz et al., 1995), which postulated strong links between the center of attention in comprehension of adjacent sentences and syntactic position and form of reference. In a detailed study of information ordering in three very different corpora, (Karamanis et al., to appear) assessed the performance of various formulations of centering. Their results were somewhat unexpected, showing that while centering transition preferences were useful, the most successful strategy for information ordering was based on avoiding rough shifts, that is, sequences of sentences that share no entities in common. This supports previous findings that such types of transitions are associated with poorly written text and can be used to improve the accuracy of automatic grading of essays based on various non-discourse features (Miltsakaki and Kukich, 2000). In a more powerful generalization of centering, Barzilay and Lapata (2008) developed a novel approach which doesn't postulate a preference for any type of transition but rather computes a set of features that capture transitions of all kinds in the text and their relative proportion. Their entity coherence features prove to be very suitable for various tasks, notably for information ordering and reading difficulty level.

Form of reference is also important in well-written text and appropriate choices lead to improved readability. Use of pronouns for reference to highly salient entities is perceived as more desirable than the use of definite noun phrases (Gordon et al., 1993; Krahmer and Theune, 2002). The syntactic forms of first mention—when an entity is first introduced in a text—differ from those of subsequent mentions (Poesio and Vieira, 1998; Nenkova and McKeown, 2003) and can be exploited for improving and predicting text coherence (Siddharthan, 2003; Nenkova and McKeown, 2003; Elsner and Charniak, 2008).

3 Data

The objective of our study is to analyze various readability factors, including discourse relations, because few empirical studies exist that directly link discourse structure with text quality. In the past, subsections of the Penn Treebank (Marcus et al., 1994) have been annotated for discourse relations (Carlson et al., 2001; Wolf and Gibson, 2005). For our study we chose to work with the newly released Penn Discourse Treebank which is the largest annotated resource which focuses exclusively on implicit local relations between adjacent sentences and explicit discourse connectives.

3.1 Discourse annotation

The Penn Discourse Treebank (Prasad et al., 2008) is a new resource with annotations of discourse connectives and their senses in the Wall Street Journal portion of the Penn Treebank (Marcus et al., 1994). All *explicit* relations (those marked with a discourse connective) are annotated. In addition, each adjacent pair of sentences within a paragraph is annotated. If there is a discourse relation, then it is marked *implicit* and annotated with one or more connectives. If there is a relation between the sentences but adding a connective would be inappropriate, it is marked *AltLex*. If the consecutive sentences are only related by entity-based coherence (Knott et al., 2001) they are annotated with *EntRel*. Otherwise, they are annotated with *NoRel*.

Besides labeling the connective, the PDTB also annotates the *sense* of each relation. The relations are organized into a hierarchy. The top level relations are Expansion, Comparison, Contingency, and Temporal. Briefly, an expansion relation means that the second clause continues the theme of the first clause, a comparison relation indicates that something in the two clauses is being compared, contingency means that there is a causal relation between the clauses, and temporal means they occur either at the same time or sequentially.

3.2 Readability ratings

We randomly selected thirty articles from the Wall Street Journal corpus that was used in both the Penn Treebank and the Penn Discourse Treebank.[1] Each article was read by at least three college students, each of whom was given unlimited time to read the texts and perform the ratings.[2] Subjects were asked the following questions:

- How well-written is this article?
- How well does the text fit together?
- How easy was it to understand?
- How interesting is this article?

For each question, they provided a rating between 1 and 5, with 5 being the best and 1 being the worst.

[1]One of the selected articles was missing from the Penn Treebank. Thus, results that do not require syntactic information (Tables 1, 2, 4, and 6) are over all thirty articles, while Tables 3, 5, and 7 report results for the twenty-nine articles with Treebank parse trees.

[2](Lapata, 2006) found that human ratings are significantly correlated with self-paced reading times, a more direct measure of processing effort which we plan to explore in future work.

After collecting the data, it turned out that most of the time subjects gave the same rating to all questions. For competent language users, we view text readability and text coherence as equivalent properties, measuring the extent to which a text is well written. Thus for all subsequent analysis, we will use only the first question ("On a scale of 1 to 5, how well written is this text?"). The score of an article was then the average of all the ratings it received. The article scores ranged from 1.5 to 4.33, with a mean of 3.2008 and a standard deviation of .7242. The median score was 3.286.

We define our task as predicting this average rating for each article. Note that this task may be more difficult than predicting reading level, as each of these articles appeared in the Wall Street Journal and thus is aimed at the same target audience. We suspected that in classifying adult text, more subtle features might be necessary.

4 Identifying correlates of text quality

4.1 Baseline measures

We first computed the Pearson correlation coefficients between the simple metrics that most traditional readability formulas use and the average human ratings. These results are shown in Table 1. We tested *the average number of characters per word*, *average number of words per sentence*, *maximum number of words per sentence*, and *article length* (F_7).[3] Article length (F_7) was the only significant baseline factor, with correlation of -0.37. Longer articles are perceived as less well-written and harder to read than shorter ones. None of the other baseline metrics were close to being significant predictors of readability.

Average Characters/Word	r = -.0859, p = .6519
Average Words/Sentence	r = .1637, p = .3874
Max Words/Sentence	r = .0866, p = .6489
F_7 text length	**r = -.3713, p = .0434**

Table 1: Baseline readability features

[3]For ease of reference, we number each non-baseline feature in the text and tables.

4.2 Vocabulary

We use a unigram language model, where the probability of an article is:

$$\prod_w P(w|M)^{C(w)} \tag{1}$$

$P(w|M)$ is the probability of word-type w according to a background corpus M, and $C(w)$ is the number of times w appears in the article.

The log likelihood of an article is then:

$$\sum_w C(w)\log(P(w|M)) \tag{2}$$

Note that this model will be biased in favor of shorter articles. Since each word has probability less than 1, the log probability of each word is less than 0, and hence including additional words decreases the log likelihood. We compensate for this by performing linear regressions with the unigram log likelihood and with the number of words in the article as an additional variable.

The question then arises as to what to use as a background corpus. We chose to experiment with two corpora: the entire Wall Street Journal corpus and a collection of general AP news, which is generally more diverse than the financial news found in the WSJ. We predicted that the NEWS vocabulary would be more representative of the types of words our readers would be familiar with. In both cases we used Laplace smoothing over the word frequencies and a stoplist.

The vocabulary features we used are *article likelihood estimated from a language model from WSJ* (F_5), and *article likelihood according to a unigram language model from NEWS* (F_6). We also combine the two likelihood features with article length, in order to get a better estimate of the language model's influence on readability independent of the length of the article.

F_5 Log likelihood, WSJ	**r = .3723, p = .0428**
F_6 Log likelihood, NEWS	**r= .4497, p = .0127**
LL with length, WSJ	**r = .3732, p = .0422**
LL with length, NEWS	**r = .6359, p = .0002**

Table 2: Vocabulary features

Both vocabulary-based features (F_5 and F_6) are significantly correlated with the readability judgments, with p-values smaller than 0.05 (see Table 2).

The correlations are positive: the more probable an article was based on its vocabulary, the higher it was generally rated. As expected, the NEWS model that included more general news stories had a higher correlation with people's judgments. When combined with the length of the article, the unigram language model from the NEWS corpus becomes very predictive of readability, with the correlation between the two as high as 0.63.

4.3 Syntactic features

Syntactic constructions affect processing difficulty and so might also affect readability judgments. We examined the four syntactic features used in (Schwarm and Ostendorf, 2005): *average parse tree height* (F_1), *average number of noun phrases per sentence* (F_2), *average number of verb phrases per sentence* (F_3), and *average number of subordinate clauses per sentence(SBARs in the Penn Treebank tagset) (F_4).* The sentence "We're talking about years ago [SBAR before anyone heard of asbestos having any questionable properties]." contains an example of an SBAR clause.

Having multiple noun phrases (entities) in each sentence requires the reader to remember more items, but may make the article more interesting. (Barzilay and Lapata, 2008) found that articles written for adults tended to contain many more entities than articles written for children. While including more verb phrases in each sentence increases the sentence complexity, adults might prefer to have related clauses explicitly grouped together.

F_1 Average Parse Tree Height	r = -.0634, p = .7439
F_2 Average Noun Phrases	r = .2189, p = .2539
F_3 **Average Verb Phrases**	**r = .4213, p = .0228**
F_4 Average SBARs	r = .3405, p = .0707

Table 3: Syntax-related features

The correlations between readability and syntactic features is shown in Table 3. The strongest correlation is that between readability and number of verb phrases (0.42). This finding is in line with prescriptive clear writing advice (Gunning, 1952; Spandel, 2004), but is to our knowledge novel in the computational linguistics literature. As (Bailin and Grafstein, 2001) point out, the sentences in (1) are easier to comprehend than the sentences in (2), even though they are longer.

(1) It was late at night, but it was clear. The stars were out and the moon was bright.

(2) It was late at night. It was clear. The stars were out. The moon was bright.

Multiple verb phrases in one sentence may be indicative of explicit discourse relations, which we will discuss further in section 4.6.

Surprisingly, the use of clauses introduced by a (possibly empty) subordinating conjunction (SBAR), are actually positively correlated (and almost approaching significance) with readability. So while for children or less educated adults these constructions might pose difficulties, they were favored by our assessors. On the other hand, the average parse tree height negatively correlated with readability as expected, but surprisingly the correlation is very weak (-0.06).

4.4 Elements of lexical cohesion

In their classic study of cohesion in English, (Halliday and Hasan, 1976) discuss the various aspects of well written discourse, including the use of cohesive devices such as pronouns, definite descriptions and topic continuity from sentence to sentence.[4] To measure the association between these features and readability rankings, we compute *the number of pronouns per sentence* (F_{11}) and *the number of definite articles per sentence* (F_{12}). In order to qualify topic continuity from sentence to sentence in the articles, we compute *average cosine similarity* (F_8), *word overlap* (F_9) and *word overlap over just nouns and pronouns* (F_{10}) between pairs of adjacent sentences[5]. Each sentence is turned into a vector of word-types, where each type's value is its tf-idf (where document frequency is computed over all the articles in the WSJ corpus). The cosine similarity metric is then:

$$\cos(s,t) = \frac{s \cdot t}{|s|\,|t|} \tag{3}$$

[4]Other cohesion building devises discussed by Halliday and Hansan include lexical reiteration and discourse relations, which we address next.

[5]Similar features have been used for automatic essay grading as well (Higgins et al., 2004).

F_8 Avr. Cosine Overlap	r = -.1012, p = .5947
F_9 Avr. Word Overlap	r = -.0531, p = .7806
F_{10} Avr. Noun+Pronoun Overlap	r = .0905, p = .6345
F_{11} Avr. # Pronouns/Sent	r = .2381, p = .2051
F_{12} Avr # Definite Articles	r = .2309, p = .2196

Table 4: Superficial measures of topic continuity and pronoun and definite description use

None of these features correlate significantly with readability as can be seen from the results in Table 4. The overlap features are particularly bad predictors of readability, with average word/cosine overlap in fact being negatively correlated with readability. The form of reference—use of pronouns and definite descriptions—exhibit a higher correlation with readability (0.23), but these values are not significant for the size of our corpus.

4.5 Entity coherence

We use the Brown Coherence Toolkit[6] to compute entity grids (Barzilay and Lapata, 2008) for each article. In each sentence, an entity is identified as the subject (S), object (O), other (X) (for example, part of a prepositional phrase), or not present (N). The probability of each transition type is computed. For example, an S-O transition occurs when an entity is the subject in one sentence then an object in the next; X-N transition occurs when an entity appears in non-subject or object position in one sentence and not present in the next, etc.[7] The entity coherence features are the *probability of each of these pairs of transitions*, for a total of 16 features (F_{17-32}; see complete results in Table 5).

None of the entity grid features are significantly correlated with the readability ratings. One very interesting result is that the proportion of S-S transitions in which the same entity was mentioned in subject position in two adjacent sentences, is negatively correlated with readability. In centering theory, this is considered the most coherent type of transition, keeping the same center of attention. Moreover, the feature most strongly correlated with readability is the S-N transition (0.31) in which the subject of one sentence does not appear at all in the following sentence. Of course, it is difficult to interpret the entity grid features one by one, since they are interdependent and probably it is the interaction of features (relative proportions of transitions) that capture overall readability patterns.

[6]http://www.cs.brown.edu/ melsner/manual.html

[7]The Brown Coherence Toolkit identifies NPs as the same entity if they have identical head nouns.

F_{17} Prob. of S-S transition	r = -.1287, p = .5059
F_{18} Prob. of S-O transition	r = -.0427, p = .8261
F_{19} Prob. of S-X transition	r = -.1450, p = .4529
F_{20} Prob. of S-N transition	r = .3116, p = .0999
F_{21} Prob. of O-S transition	r = .1131, p = .5591
F_{22} Prob. of O-O transition	r = .0825, p = .6706
F_{23} Prob. of O-X transition	r = .0744, p = .7014
F_{24} Prob. of O-N transition	r = .2590, p = .1749
F_{25} Prob. of X-S transition	r = .1732, p = .3688
F_{26} Prob. of X-O transition	r = .0098, p = .9598
F_{27} Prob. of X-X transition	r = -.0655, p = .7357
F_{28} Prob. of X-N transition	r = .1319, p = .4953
F_{29} Prob. of N-S transition	r = .1898, p = .3242
F_{30} Prob. of N-O transition	r = .2577, p = .1772
F_{31} Prob. of N-X transition	r = .1854, p = .3355
F_{32} Prob. of N-N transition	r = -.2349, p = .2200

Table 5: Linear correlation between human readability ratings and entity coherence.

4.6 Discourse relations

Discourse relations are believed to be a major factor in text coherence. We computed another language model which is over discourse relations instead of words. We treat each text as a bag of relations rather than a bag of words. Each relation is annotated for both its sense and how it is realized (implicit or explicit). For example, one text might contain {Implicit Comparison, Explicit Temporal, NoRel}. We computed the probability of each of our articles according to a multinomial model, where the probability of a text with n relation tokens and k relation types is:

$$P(n)\frac{n!}{x_1!...x_k!}p_1^{x_1}...p_k^{x_k} \quad (4)$$

$P(n)$ is the probability of an article having length n, x_i is the number of times relation i appeared, and p_i is the probability of relation i based on the Penn Discourse Treebank. $P(n)$ is the maximum likelihood estimation of an article having n discourse relations based on the entire Penn Discourse Treebank (the number of articles with exactly n discourse relations, divided by the total number of articles).

The *log likelihood of an article based on its discourse relations* (F_{13}) feature is defined as:

$$\log(P(n)) + \log(n!) + \sum_{i=1}^{k} (x_i \log(p_i) - \log(x_i!)) \quad (5)$$

The multinomial distribution is particularly suitable, because it directly incorporates length, which significantly affects readability as we discussed earlier. It also captures patterns of relative frequency of relations, unlike the simpler unigram model. Note also that this equation has an advantage over the unigram model that was not present for vocabulary. While every article contains at least one word, some articles do not contain any discourse relations. Since the PDTB annotated all explicit relations and relations between adjacent sentences in a paragraph, an article with no discourse connectives and only single sentence paragraphs would not contain any annotated discourse relations. Under the unigram model, these articles' probabilities cannot be computed. Under the multinomial model, the probability of an article with zero relations is estimated as $Pr(N = 0)$, which can be calculated from the corpus.

As in the case of vocabulary features, the presence of more relations will lead to overall lower probabilities so we also consider the *number of discourse relations* (F_{14}) and *the log likelihood combined with the number of relations* as features. In order to isolate the effect of the type of discourse relation (explicitly expressed by a discourse connective such as "because" or "however" versus implicitly expressed by adjacency), we also compute multinomial model features for the *explicit discourse relations* (F_{15}) and over just the *implicit discourse relations* (F_{16}).

F_{13} **LogL of discourse rels**	**r = .4835, p = .0068**
F_{14} # of discourse relations	r = -.2729, p = .1445
LogL of rels with # of rels	**r = .5409, p = .0020**
# of relations with # of words	**r = .3819, p = .0373**
F_{15} Explicit relations only	r = .1528, p = .4203
F_{16} Implicit relations only	r = .2403, p = .2009

Table 6: Discourse features

The likelihood of discourse relations in the text under a multinomial model is very highly and significantly correlated with readability ratings, especially after text length is taken into account. Correlations are 0.48 and 0.54 respectively. The probability of the explicit relations alone is not a sufficiently strong indicator of readability. This fact is disappointing as the explicit relations can be identified much more easily in unannotated text (Pitler et al., 2008). Note that the sequence of just the implicit relations is also not sufficient. This observation implies that the proportion of explicit and implicit relations may be meaningful but we leave the exploration of this issue for later work.

4.7 Summary of findings

So far, we introduced six classes of factors that have been discussed in the literature as readability correlates. Through statistical tests of associations we identified the individual factors significantly correlated with readability ratings. These are, in decreasing order of association strength:

LogL of Discourse Relations (r = .4835)
LogL, NEWS (r= .4497)
Average Verb Phrases (.4213)
LogL, WSJ (r = .3723)
Number of words (r = -.3713)

Vocabulary and discourse relations are the strongest predictors of readability, followed by average number of verb phrases and length of the text. This empirical confirmation of the significance of discourse relations as a readability factor is novel for the computational linguistics literature. Note though that for our work we use oracle discourse annotations directly from the PDTB and no robust systems for automatic discourse annotation exist today.

The significance of the average number of verb phrases as a readability predictor is somewhat surprising but intriguing. It would lead to reexamination of the role of verbs/predicates in written text, which we also plan to address in future work. None of the other factors showed significant association with readability ratings, even though some correlations had relatively large positive values.

5 Combining readability factors

In this section, we turn to the question of how the combination of various factors improves the prediction of readability. We use the **leaps** package in R to find the best subset of features for linear regression, for subsets of size one to eight. We use the squared multiple correlation coefficient (R^2) to assess the effectiveness of predictions. R^2 is the proportion of variance in readability ratings explained by the model. If the model predicts readability perfectly, $R^2 = 1$, and if the model has no predictive capability, $R^2 = 0$.

F_{13}, $R^2 = 0.2662$
$F_6 + F_7$, $R^2 = 0.4351$
$F_6 + F_7 + F_{13}$, $R^2 = 0.5029$
$F_6 + F_7 + F_{13} + F_{14}$, $R^2 = 0.6308$
$F_1 + F_6 + F_7 + F_{10} + F_{13}$, $R^2 = 0.6939$
$F_1 + F_6 + F_7 + F_{10} + F_{13} + F_{23}$, $R^2 = 0.7316$
$F_1 + F_6 + F_7 + F_{10} + F_{13} + F_{22} + F_{23}$, $R^2 = 0.7557$
$F_1 + F_6 + F_7 + F_{10} + F_{11} + F_{13} + F_{19} + F_{30}$, $R^2 = 0.776$.

The linear regression results confirm the expectation that the combination of different factors is a rather complex issue. As expected, discourse, vocabulary and length which were the significant individual factors appear in the best model for each feature set size. Their combination gives the best result for regression with three predictors, and they explain half of the variance in readability ratings, $R^2 = 0.5029$.

But the other individually significant feature, average number of verb phrases per sentence (F_3) never appears in the best models. Instead, F_1—the depth of the parse tree—appears in the best model with more than four features.

Also unexpectedly, two of the superficial cohesion features appear in the larger models: F_{10} is the average word overlap over nouns and pronouns and F_{11} is the average number of pronouns per sentence. Entity grid features also make their way into the best models when more features are used for prediction: S-X, O-O, O-X, N-O transitions (F_{19}, F_{22}, F_{23}, F_{30}).

6 Readability as ranking

In this section we consider the problem of pairwise ranking of text readability. That is, rather than trying to predict the readability of a single document, we consider pairs of documents and predict which one is better. This task may in fact be the more natural one, since in most applications the main concern is with the relative quality of articles rather than their absolute scores. This setting is also beneficial in terms of data use, because each pair of articles with different average readability scores now becomes a data point for the classification task.

We thus create a classification problem: given two articles, is article 1 more readable than article 2? For each pair of texts whose readability ratings on the 1 to 5 scale differed by at least 0.5, we form one data point for the ranking problem, resulting in 243 examples. The predictors are the differences between the two articles' features. For classification, we used WEKA's linear support vector implementation (SMO) and performance was evaluated using 10-fold cross-validation.

Features	**Accuracy**
None (Majority Class)	50.21%
ALL	88.88%
log_l_discourse_rels	77.77%
number_discourse_rels	74.07%
N-O transition	70.78%
O-N transition	69.95%
Avg_VPs_sen	69.54%
log_l_NEWS	66.25%
number_of_words	65.84%
Grid only	79.42%
Discourse only	77.36%
Syntax only	74.07%
Vocab only	66.66%
Length only	65.84%
Cohesion only	64.60%
no cohesion	89.30%
no vocab	88.88%
no length	88.47%
no discourse	88.06%
no grid	84.36%
no syntax	82.71%

Table 7: SVM prediction accuracy, linear kernel

The classification results are shown in Table 7. When all features are used for prediction, the accuracy is high, 88.88%. The length of the article can serve as a baseline feature—longer articles are ranked lower by the assessors, so this feature can be taken as baseline indicator of readability. Only six features used by themselves lead to accuracies higher than the length baseline. These results indicate that the most important individual factors in the readability ranking task, in decreasing order of importance, are log likelihood of discourse relations, number of discourse relations, N-O transitions, O-N transitions, average number of VPs per sentence and text probability under a general language model.

In terms of classes of features, the 16 entity grid features perform the best, leading to an accuracy of 79.41%, followed by the combination of the four discourse features (77.36%), and syntax features (74.07%). This is evidence for the fact that there is a complex interplay between readability factors: the entity grid factors which individually have very weak correlation with readability combine well, while adding the three additional discourse features to the likelihood of discourses relations actually worsens performance slightly. Similar indication for interplay between features is provided by the class ablation classification results, in which classes of features are removed. Surprisingly, removing syntactic features causes the biggest deterioration in performance, a drop in accuracy from 88.88% to 82.71%. The removal of vocabulary, length, or discourse features has a minimal negative impact on performance, while removing the cohesion features actually boosts performance.

7 Conclusion

We have investigated which linguistic features correlate best with readability judgments. While surface measures such as the average number of words per sentence or the average number of characters per word are not good predictors, there exist syntactic, semantic, and discourse features that do correlate highly. The average number of verb phrases in each sentence, the number of words in the article, the likelihood of the vocabulary, and the likelihood of the discourse relations all are highly correlated with humans' judgments of how well an article is written.

While using any one out of syntactic, lexical, coherence, or discourse features is substantally better than the baseline surface features on the discrimination task, using a combination of entity coherence and discourse relations produces the best performance.

8 Acknowledgments

This work was partially supported by an Integrative Graduate Education and Research Traineeship grant from National Science Foundation (NSFIGERT 0504487) and by NSF Grant IIS-07-05671. We thank Aravind Joshi, Bonnie Webber, and the anonymous reviewers for their many helpful comments.

References

Y. Attali and J. Burstein. 2006. Automated essay scoring with e-rater v.2. *The Journal of Technology, Learning and Assessment*, 4(3).

A. Bailin and A. Grafstein. 2001. The linguistic assumptions underlying readability formulae a critique. *Language and Communication*, 21(3):285–301.

R. Barzilay and M. Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

L. Carlson, D. Marcu, and M. E. Okurowski. 2001. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Proceedings of the Second SIGdial Workshop*, pages 1–10.

M. Coleman and TL Liau. 1975. A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, 60(2):283–284.

K. Collins-Thompson and J. Callan. 2004. A language modeling approach to predicting reading difficulty. In *Proceedings of HLT/NAACL'04*.

Noemie Elhadad and Komal Sutaria. 2007. Mining a lexicon of technical terms and lay equivalents. In *Biological, translational, and clinical language processing*, pages 49–56, Prague, Czech Republic. Association for Computational Linguistics.

M. Elsner and E. Charniak. 2008. Coreference-inspired coherence modeling. In *Proceedings of ACL-HLT'08, (short paper)*.

E. Gibson. 1998. Linguistic complexity: locality of syntactic dependencies. *Cognition*, 68:1–76.

P. Gordon, B. Grosz, and L. Gilliom. 1993. Pronouns, names, and the centering of attention in discourse. *Cognitive Science*, 17:311–347.

B. Grosz, A. Joshi, and S. Weinstein. 1995. Centering: a framework for modelling the local coherence of discourse. *Computational Linguistics*, 21(2):203–226.

Robert Gunning. 1952. *The technique of clear writing*. McGraw-Hill; Fouth Printing edition.

Michael A.K. Halliday and Ruqaiya Hasan. 1976. *Cohesion in English*. Longman Group Ltd, London, U.K.

M. Heilman, K. Collins-Thompson, J. Callan, and M. Eskenazi. 2007. Combining Lexical and Grammatical Features to Improve Readability Measures for First and Second Language Texts. *Proceedings of NAACL HLT*, pages 460–467.

D. Higgins, J. Burstein, D. Marcu, and C. Gentile. 2004. Evaluating multiple aspects of coherence in student essays. In *Proceedings of HLT/NAACL'04*.

N. Karamanis, M. Poesio, C. Mellish, and J. Oberlander. (to appear). Evaluating centering for information ordering using corpora. *Computational Linguistics*.

JP Kincaid. 1975. Derivation of New Readability Formulas (Automated Readability Index, Fog Count and Flesch Reading Ease Formula) for Navy Enlisted Personnel.

A. Knott, J. Oberlander, M. ODonnell, and C. Mellish. 2001. Beyond elaboration: The interaction of relations and focus in coherent text. *Text representation: linguistic and psycholinguistic aspects*, pages 181–196.

E. Krahmer and M. Theune. 2002. Efficient context-sensitive generation of referring expressions. In K. van Deemter and R. Kibble, editors, *Information Sharing: Reference and Presupposition in Language Generation and Interpretation*, pages 223–264. CSLI Publications.

M. Lapata. 2006. Automatic evaluation of information ordering: Kendalls tau. *Computational Linguistics*, 32(4):471–484.

W. Mann and S. Thompson. 1988. Rhetorical structure theory: Towards a functional theory of text organization. *Text*, 8.

M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1994. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

G.H. McLaughlin. 1969. SMOG grading: A new readability formula. *Journal of Reading*, 12(8):639–646.

E. Miltsakaki and K. Kukich. 2000. The role of centering theory's rough-shift in the teaching and evaluation of writing skills. In *Proceedings of ACL'00*, pages 408–415.

A. Nenkova and K. McKeown. 2003. References to named entities: a corpus study. In *Proceedings of HLT/NAACL 2003 (short paper)*.

E. Pitler, M. Raghupathy, H. Mehta, A. Nenkova, A. Lee, and A. Joshi. 2008. Easily identifiable discourse relations. In *Coling 2008: Companion volume: Posters and Demonstrations*, pages 85–88, Manchester, UK, August.

M. Poesio and R. Vieira. 1998. A corpus-based investigation of definite description use. *Computational Linguistics*, 24(2):183–216.

R. Prasad, N. Dinesh, A. Lee, E. Miltsakaki, L. Robaldo, A. Joshi, and B. Webber. 2008. The penn discourse treebank 2.0. In *Proceedings of LREC'08*.

KA Schriver. 1989. Evaluating text quality: the continuum from text-focused toreader-focused methods. *Professional Communication, IEEE Transactions on*, 32(4):238–255.

S. Schwarm and M. Ostendorf. 2005. Reading level assessment using support vector machines and statistical language models. In *Proceedings of ACL'05*, pages 523–530.

L. Si and J. Callan. 2001. A statistical model for scientific readability. *Proceedings of the tenth international conference on Information and knowledge management*, pages 574–576.

A. Siddharthan. 2003. *Syntactic simplification and Text Cohesion*. Ph.D. thesis, University of Cambridge, UK.

V. Spandel. 2004. *Creating writers through 6-trait writing assessment and instruction*. Allyn & Bacon.

F. Wolf and E. Gibson. 2005. Representing discourse coherence: A corpus-based study. *Computational Linguistics*, 31(2):249–288.

Syntactic Constraints on Paraphrases Extracted from Parallel Corpora

Chris Callison-Burch
Center for Language and Speech Processing
Johns Hopkins University
Baltimore, Maryland
ccb@cs.jhu.edu

Abstract

We improve the quality of paraphrases extracted from parallel corpora by requiring that phrases and their paraphrases be the same syntactic type. This is achieved by parsing the English side of a parallel corpus and altering the phrase extraction algorithm to extract phrase labels alongside bilingual phrase pairs. In order to retain broad coverage of non-constituent phrases, complex syntactic labels are introduced. A manual evaluation indicates a 19% absolute improvement in paraphrase quality over the baseline method.

1 Introduction

Paraphrases are alternative ways of expressing the same information. Being able to identify or generate paraphrases automatically is useful in a wide range of natural language applications. Recent work has shown how paraphrases can improve question answering through query expansion (Riezler et al., 2007), automatic evaluation of translation and summarization by modeling alternative lexicalization (Kauchak and Barzilay, 2006; Zhou et al., 2006; Owczarzak et al., 2006), and machine translation both by dealing with out of vocabulary words and phrases (Callison-Burch et al., 2006) and by expanding the set of reference translations for minimum error rate training (Madnani et al., 2007). While all applications require the preservation of meaning when a phrase is replaced by its paraphrase, some additionally require the resulting sentence to be grammatical.

In this paper we examine the effectiveness of placing syntactic constraints on a commonly used paraphrasing technique that extracts paraphrases from parallel corpora (Bannard and Callison-Burch, 2005). The paraphrasing technique employs various aspects of phrase-based statistical machine translation including phrase extraction heuristics to obtain bilingual phrase pairs from word alignments. English phrases are considered to be potential paraphrases of each other if they share a common foreign language phrase among their translations. Multiple paraphrases are frequently extracted for each phrase and can be ranked using a paraphrase probability based on phrase translation probabilities.

We find that the quality of the paraphrases that are generated in this fashion improves significantly when they are required to be the same syntactic type as the phrase that they are paraphrasing. This constraint:

- Eliminates a trivial but pervasive error that arises from the interaction of unaligned words with phrase extraction heuristics.
- Refines the results for phrases that can take on different syntactic labels.
- Applies both to phrases which are linguistically coherent and to arbitrary sequences of words.
- Results in much more grammatical output when phrases are replaced with their paraphrases.

A thorough manual evaluation of the refined paraphrasing technique finds a 19% absolute improve-

Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, pages 196–205,
Honolulu, October 2008. ©2008 Association for Computational Linguistics

ment in the number of paraphrases that are judged to be correct.

This paper is structured as follows: Section 2 describes related work in syntactic constraints on phrase-based SMT and work utilizing syntax in paraphrase discovery. Section 3 details the problems with extracting paraphrases from parallel corpora and our improvements to the technique. Section 4 describes our experimental design and evaluation methodology. Section 5 gives the results of our experiments, and Section 6 discusses their implications.

2 Related work

A number of research efforts have focused on employing syntactic constraints in statistical machine translation. Wu (1997) introduced the inversion transduction grammar formalism which treats translation as a process of parallel parsing of the source and target language via a synchronized grammar. The synchronized grammar places constraints on which words can be aligned across bilingual sentence pairs. To achieve computational efficiency, the original proposal used only a single non-terminal label rather than a linguistic grammar.

Subsequent work used more articulated parses to improve alignment quality by applying cohesion constraints (Fox, 2002; Lin and Cherry, 2002). If two English phrases are in disjoint subtrees in the parse, then the phrasal cohesion constraint prevents them from being aligned to overlapping sequences in the foreign sentence. Other recent work has incorporated constituent and dependency subtrees into the translation rules used by phrase-based systems (Galley et al., 2004; Quirk et al., 2005). Phrase-based rules have also been replaced with synchronous context free grammars (Chiang, 2005) and with tree fragments (Huang and Knight, 2006).

A number of techniques for generating paraphrases have employed syntactic information, either in the process of extracting paraphrases from monolingual texts or in the extracted patterns themselves. Lin and Pantel (2001) derived paraphrases based on the distributional similarity of paths in dependency trees. Barzilay and McKeown (2001) incorporated part-of-speech information and other morphosyntactic clues into their co-training algorithm. They extracted paraphrase patterns that incorporate this information. Ibrahim et al. (2003) generated structural paraphrases capable of capturing long-distance dependencies. Pang et al. (2003) employed a syntax-based algorithm to align equivalent English sentences by merging corresponding nodes in parse trees and compressing them down into a word lattice.

Perhaps the most closely related work is a recent extension to Bannard and Callison-Burch's paraphrasing method. Zhao et al. (2008b) extended the method so that it is capable of generating richer paraphrase patterns that include part-of-speech slots, rather than simple lexical and phrasal paraphrases. For example, they extracted patterns such as *consider NN* $\rightarrow$ *take NN into consideration*. To accomplish this, Zhao el al. used dependency parses on the English side of the parallel corpus. Their work differs from the work presented in this paper because their syntactic constraints applied to slots within paraphrase patters, and our constraints apply to the paraphrases themselves.

3 Paraphrasing with parallel corpora

Bannard and Callison-Burch (2005) extract paraphrases from bilingual parallel corpora. They give a probabilistic formation of paraphrasing which naturally falls out of the fact that they use techniques from phrase-based statistical machine translation:

$$\hat{e_2} = \arg\max_{e_2 : e_2 \neq e_1} p(e_2|e_1) \tag{1}$$

where

$$\begin{aligned} p(e_2|e_1) &= \sum_f p(f|e_1)p(e_2|f, e_1) && (2) \\ &\approx \sum_f p(f|e_1)p(e_2|f) && (3) \end{aligned}$$

Phrase translation probabilities $p(f|e_1)$ and $p(e_2|f)$ are commonly calculated using maximum likelihood estimation (Koehn et al., 2003):

$$p(f|e) = \frac{count(e, f)}{\sum_f count(e, f)} \tag{4}$$

where the counts are collected by enumerating all bilingual phrase pairs that are consistent with the

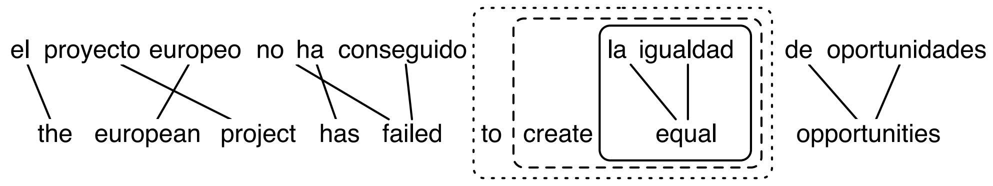

el proyecto europeo no ha conseguido
la igualdad
de oportunidades
the european project has failed
to create equal
opportunities

Figure 1: The interaction of the phrase extraction heuristic with unaligned English words means that the Spanish phrase *la igualdad* aligns with *equal*, *create equal*, and *to create equal*.

word alignments for sentence pairs in a bilingual parallel corpus. Various phrase extraction heuristics are possible. Och and Ney (2004) defined consistent bilingual phrase pairs as follows:

$$\begin{aligned} BP(f_1^J, e_1^I, A) = \{(f_j^{j+m}, e_i^{i+n}) : \\ \forall(i', j') \in A : j \leq j' \leq j+m \leftrightarrow i \leq i' \leq i+n \\ \wedge \exists(i', j') \in A : j \leq j' \leq j+m \wedge \leftrightarrow i \leq i' \leq i+n\} \end{aligned}$$

where f_1^J is a foreign sentence, e_1^I is an English sentence and A is a set of word alignment points.

The heuristic allows unaligned words to be included at the boundaries of the source or target language phrases. For example, when enumerating the consistent phrase pairs for the sentence pair given in Figure 1, *la igualdad* would align not only to *equal*, but also to *create equal*, and *to create equal*. In SMT these alternative translations are ranked by the translation probabilities and other feature functions during decoding.

The interaction between the phrase extraction heuristic and unaligned words results in an undesirable effect for paraphrasing. By Bannard and Callison-Burch's definition, *equal*, *create equal*, and *to create equal* would be considered paraphrases because they are aligned to the same foreign phrase. Tables 1 and 2 show how sub- and super-phrases can creep into the paraphrases: *equal* can be paraphrased as *equal rights* and *create equal* can be paraphrased as *equal*. Obviously when e_2 is substituted for e_1 the resulting sentence will generally be ungrammatical. The first case could result in *equal equal rights*, and the second would drop the verb.

This problem is pervasive. To test its extent we attempted to generate paraphrases for 900,000 phrases using Bannard and Callison-Burch's method trained on the Europarl corpora (as described in Section 4). It generated a total of 3.7 million paraphrases for 400,000 phrases in the list.[1] We observed that 34% of the paraphrases (excluding the phrase itself) were super- or sub-strings of the original phrase. The most probable paraphrase was a super- or sub-string of the phrase 73% of the time.

equal			
equal	.35	equally	.02
same	.07	the	.02
equality	.03	fair	.01
equals	.02	equal rights	.01

Table 1: The baseline method's paraphrases of *equal* and their probabilities (excluding items with $p < .01$).

create equal			
create equal	.42	same	.03
equal	.06	created	.02
to create a	.05	conditions	.02
create	.04	playing	.02
to create equality	.03	creating	.01

Table 2: The baseline's paraphrases of *create equal*. Most are clearly bad, and the most probable $e_2 \neq e_1$ is a substring of e_1.

There are a number of strategies that might be adopted to alleviate this problem:

- Bannard and Callison-Burch (2005) rank their paraphrases with a language model when the paraphrases are substituted into a sentence.

- Bannard and Callison-Burch (2005) sum over multiple parallel corpora C to reduce the problems associated with systematic errors in the

[1]The remaining 500,000 phrases could not be paraphrased either because $e_2 \neq e_1$ or because they were not consistently aligned to any foreign phrases.

word alignments in one language pair:

$$\hat{e_2} = \arg\max_{e_2} \sum_{c \in C} \sum_{f} p(f|e_1)p(e_2|f) \qquad (5)$$

- We could change the phrase extraction heuristic's treatment of unaligned words, or we could attempt to ensure that we have fewer unaligned items in our word alignments.
- The paraphrase criterion could be changed from being $e_2 \neq e_1$ to specifying that $e2$ is not sub- or super-string of $e1$.

In this paper we adopt a different strategy. The essence of our strategy is to constrain paraphrases to be the same syntactic type as the phrases that they are paraphrasing. Syntactic constraints can apply in two places: during phrase extraction and when substituting paraphrases into sentences. These are described in sections 3.1 and 3.2.

3.1 Syntactic constraints on phrase extraction

When we apply syntactic constraints to the phrase extraction heuristic, we change how bilingual phrase pairs are enumerated and how the component probabilities of the paraphrase probability are calculated.

We use the syntactic type s of e_1 in a refined version of the paraphrase probability:

$$\hat{e_2} = \arg\max_{e_2 : e_2 \neq e_1 \wedge s(e_2) = s(e_1)} p(e_2|e_1, s(e_1)) \qquad (6)$$

where $p(e_2|e_1, s(e_1))$ can be approximated as:

$$\sum_{c \in C} \frac{\sum_f p(f|e_1, s(e_1))p(e_2|f, s(e_1))}{|C|} \qquad (7)$$

We define a new phrase extraction algorithm that operates on an English parse tree P along with foreign sentence f_1^J, English sentence e_1^I, and word alignment A. We dub this SBP for *syntactic bilingual phrases*:

$$\begin{aligned} SBP(f_1^J, e_1^I, A, P) = \{(f_j^{j+m}, e_i^{i+n}, s(e_i^{i+n})) : \\ \forall (i', j') \in A : j \leq j' \leq j+m \leftrightarrow i \leq i' \leq i+n \\ \wedge \exists (i', j') \in A : j \leq j' \leq j+m \wedge \leftrightarrow i \leq i' \leq i+n \\ \wedge \exists \text{ subtree } \in P \text{ with label } s \text{ spanning words } (i, i+n)\} \end{aligned}$$

equal

JJ	equal	.60	similar	.02
	same	.14	equivalent	.01
	fair	.02		
ADJP	equal	.79	the same	.01
	necessary	.02	equal in law	.01
	similar	.02	equivalent	.01
	identical	.02		

Table 3: Syntactically constrained paraphrases for *equal* when it is labeled as an adjective or adjectival phrase.

The SBP phrase extraction algorithm produces tuples containing a foreign phrase, an English phrase and a syntactic label (f, e, s). After enumerating these for all phrase pairs in a parallel corpus, we can calculate $p(f|e_1, s(e_1))$ and $p(e_2|f, s(e_1))$ as:

$$\begin{aligned} p(f|e_1, s(e_1)) &= \frac{count(f, e_1, s(e_1))}{\sum_f count(f, e_1, s(e_1))} \\ p(e_2|f, s(e_1)) &= \frac{count(f, e_2, s(e_1))}{\sum_{e_2} count(f, e_2, s(e_1))} \end{aligned}$$

By redefining the probabilities in this way we partition the space of possible paraphrases by their syntactic categories.

In order to enumerate all phrase pairs with their syntactic labels we need to parse the English side of the parallel corpus (but not the foreign side). This limits the potential applicability of our refined paraphrasing method to languages which have parsers.

Table 3 gives an example of the refined paraphrases for *equal* when it occurs as an adjective or adjectival phrase. Note that most of the paraphrases that were possible under the baseline model (Table 1) are now excluded. We no longer get the noun *equality*, the verb *equals*, the adverb *equally*, the determier *the* or the NP *equal rights*. The paraphrases seem to be higher quality, especially if one considers their fidelity when they replace the original phrase in the context of some sentence.

We tested the rate of paraphrases that were sub- and super-strings when we constrain paraphrases based on non-terminal nodes in parse trees. The percent of the best paraphrases being substrings dropped from 73% to 24%, and the overall percent of paraphrases subsuming or being subsumed by the original phrase dropped from 34% to 12%. However, the number of phrases for which we were able

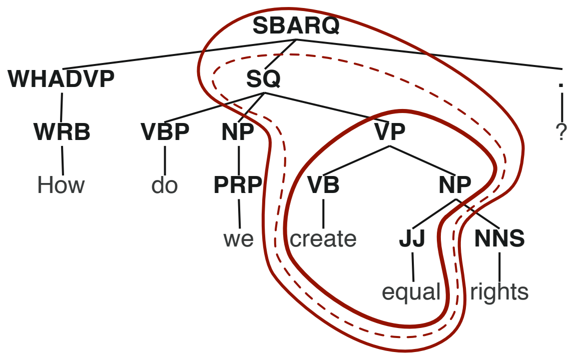


Figure 2: In addition to extracting phrases that are dominated by a node in the parse tree, we also generate labels for non-syntactic constituents. Three labels are possible for *create equal*.

to generated paraphrases dropped from 400,000 to 90,000, since we limited ourselves to phrases that were valid syntactic constituents. The number of unique paraphrases dropped from several million to 800,000.

The fact that we are able to produce paraphrases for a much smaller set of phrases is a downside to using syntactic constraints as we have initially proposed. It means that we would not be able to generate paraphrases for phrases such as *create equal*. Many NLP tasks, such as SMT, which could benefit from paraphrases require broad coverage and may need to paraphrases for phrases which are not syntactic constituents.

Complex syntactic labels

To generate paraphrases for a wider set of phrases, we change our phrase extraction heuristic again so that it produces phrase pairs for arbitrary spans in the sentence, including spans that aren't syntactic constituents. We assign every span in a sentence a syntactic label using CCG-style notation (Steedman, 1999), which gives a syntactic role with elements missing on the left and/or right hand sides.

$$\begin{aligned} SBP(f_1^J, e_1^I, A, P) = \{(f_j^{j+m}, e_i^{i+n}, s) : \\ \forall (i', j') \in A : j \le j' \le j+m \leftrightarrow i \le i' \le i+n \\ \wedge \exists (i', j') \in A : j \le j' \le j+m \wedge \leftrightarrow i \le i' \le i+n \\ \wedge \exists s \in \mathit{CCG\text{-}labels}(e_i^{i+n}, P)\} \end{aligned}$$

The function $\mathit{CCG\text{-}labels}$ describes the set of CCG-labels for the phrase spanning positions i to $i+n$ in a parse tree P. It generates three complex syntactic labels for the non-syntactic constituent phrase *create equal* in the parse tree given in Figure 2:

1. VP/(NP/NNS) – This label corresponds to the innermost circle. It indicates that *create equal* is a verb phrase missing a noun phrase to its right. That noun phrase in turn missing a plural noun (NNS) to its right.

2. SQ\VBP NP/(VP/(NP/NNS)) – This label corresponds to the middle circle. It indicates that *create equal* is an SQ missing a VBP and a NP to its left, and the complex VP to its right.

3. SBARQ\WHADVP (SQ\VBP NP/(VP/(NP/NNS)))/. – This label corresponds to the outermost circle. It indicates that *create equal* is an SBARQ missing a WHADVP and the complex SQ to its left, and a punctuation mark to its right.

We can use these complex labels instead of atomic non-terminal symbols to handle non-constituent phrases. For example, Table 4 shows the paraphrases and syntactic labels that are generated for the non-constituent phrase *create equal*. The paraphrases are significantly better than the paraphrases generated for the phrase by the baseline method (refer back to Table 2).

The labels shown in the figure are a fraction of those that can be derived for the phrase in the parallel corpus. Each of these corresponds to a different

create equal		
VP/(NP/NNS)	create equal	.92
	creating equal	.08
VP/(NP/NNS) PP	create equal	.96
	promote equal	.03
	establish fair	.01
VP/(NP/NNS) PP PP	create equal	.80
	creating equal	.10
	provide equal	.06
	create genuinely fair	.04
VP/(NP/(NP/NN) PP)	create equal	.83
	create a level playing	.17
VP/(NP/(NP/NNS) PP)	create equal	.83
	creating equal	.17

Table 4: Paraphrases and syntactic labels for the non-constituent phrase *create equal*.

syntactic context, and each has its own set of associated paraphrases.

We increase the number of phrases that are paraphrasable from the 90,000 in our initial definition of SBP to 250,000 when we use complex CCG labels. The number of unique paraphrases increases from 800,000 to 3.5 million, which is nearly as many paraphrases that were produced by the baseline method for the sample.

3.2 Syntactic constraints when substituting paraphrases into a test sentence

In addition to applying syntactic constraints to our phrase extraction algorithm, we can also apply them when we substitute a paraphrase into a sentence. To do so, we limit the paraphrases to be the same syntactic type as the phrase that it is replacing, based on the syntactic labels that are derived from the phrase tree for a test sentence. Since each phrase normally has a set of different CCG labels (instead of a single non-termal symbol) we need a way of choosing which label to use when applying the constraint.

There are several different possibilities for choosing among labels. We could simultaneously choose the best paraphrase and the best label for the phrase in the parse tree of the test sentence:

$$\hat{e_2} = \arg\max_{e_2:e_2\neq e_1} \arg\max_{s\in CCG\text{-}labels(e_1,P)} p(e_2|e_1,s) \tag{8}$$

Alternately, we could average over all of the labels that are generated for the phrase in the parse tree:

$$\hat{e_2} = \arg\max_{e_2:e_2\neq e_1} \sum_{s\in CCG\text{-}labels(e_1,P)} p(e_2|e_1,s) \tag{9}$$

The potential drawback of using Equations 8 and 9 is that the CCG labels for a particular sentence significantly reduces the paraphrases that can be used. For instance, VP/(NP/NNS) is the only label for the paraphrases in Table 4 that is compatible with the parse tree given in Figure 2.

Because the CCG labels for a given sentence are so specific, many times there are no matches. Therefore we also investigated a looser constraint. We choose the highest probability paraphrase with any label (i.e. the set of labels extracted from all parse trees in our parallel corpus):

$$\hat{e_2} = \arg\max_{e_2:e_2\neq e_1} \arg\max_{s\in\cap_{\forall T \text{ in } C} CCG\text{-}labels(e_1,T)} p(e_2|e_1,s) \tag{10}$$

Equation 10 only applies syntactic constraints during phrase extraction and ignores them during substitution.

In our experiments, we evaluate the quality of the paraphrases that are generated using Equations 8, 9 and 10. We compare their quality against the Bannard and Callison-Burch (2005) baseline.

4 Experimental design

We conducted a manual evaluation to evaluate paraphrase quality. We evaluated whether paraphrases retained the meaning of their original phrases and whether they remained grammatical when they replaced the original phrase in a sentence.

4.1 Training materials

Our paraphrase model was trained using the Europarl corpus (Koehn, 2005). We used ten parallel corpora between English and (each of) Danish, Dutch, Finnish, French, German, Greek, Italian, Portuguese, Spanish, and Swedish, with approximately 30 million words per language for a total of 315 million English words. Automatic word alignments were created for these using Giza++ (Och and Ney, 2003). The English side of each parallel corpus was parsed using the Bikel parser (Bikel, 2002). A total of 1.6 million unique sentences were parsed. A trigram language model was trained on these English sentences using the SRI language modeling toolkit (Stolcke, 2002).

The paraphrase model and language model for the Bannard and Callison-Burch (2005) baseline were trained on the same data to ensure a fair comparison.

4.2 Test phrases

The test set was the English portion of test sets used in the shared translation task of the ACL-2007 Workshop on Statistical Machine Translation (Callison-Burch et al., 2007). The test sentences were also parsed with the Bikel parser.

The phrases to be evaluated were selected such that there was an even balance of phrase lengths (from one word long up to five words long), with half of the phrases being valid syntactic constituents and half being arbitrary sequences of words. 410 phrases were selected at random for evaluation. 30 items were excluded from our results subsequent to evaluation on the grounds that they consisted solely of punctuation and stop words like determiners, prepositions and pronouns. This left a total of 380 unique phrases.

4.3 Experimental conditions

We produced paraphrases under the following eight conditions:

1. **Baseline** – The paraphrase probability defined by Bannard and Callison-Burch (2005). Calculated over multiple parallel corpora as given in Equation 5. Note that under this condition the best paraphrase is the same for each occurrence of the phrase irrespective of which sentence it occurs in.

2. **Baseline + LM** – The paraphrase probability (as above) combined with the language model probability calculated for the sentence with the phrase replaced with the paraphrase.

3. **Extraction Constraints** – This condition selected the best paraphrase according to Equation 10. It chooses the single best paraphrase over all labels. Conditions 3 and 5 only apply the syntactic constraints at the phrase extraction stage, and do not require that the paraphrase have the same syntactic label as the phrase in the sentence that it is being subtituted into.

4. **Extraction Constraints + LM** – As above, but the paraphrases are also ranked with a language model probability.

5. **Substitution Constraints** – This condition corresponds to Equation 8, which selects the highest probability paraphrase which matches at least one of the syntactic labels of the phrase in the test sentence. Conditions 5–8 apply the syntactic constraints both and the phrase extraction and at the substitution stages.

6. **Syntactic Constraints + LM** – As above, but including a language model probability as well.

7. **Averaged Substitution Constraints** – This condition corresponds to Equation 9, which averages over all of the syntactic labels for the phrase in the sentence, instead of choosing the single one which maximizes the probability.

	MEANING
5	All of the meaning of the original phrase is retained, and nothing is added
4	The meaning of the original phrase is retained, although some additional information may be added but does not transform the meaning
3	The meaning of the original phrase is retained, although some information may be deleted without too great a loss in the meaning
2	Substantial amount of the meaning is different
1	The paraphrase doesn't mean anything close to the original phrase
	GRAMMAR
5	The sentence with the paraphrase inserted is perfectly grammatical
4	The sentence is grammatical, but might sound slightly awkward
3	The sentence has an agreement error (such as between its subject and verb, or between a plural noun and singular determiner)
2	The sentence has multiple errors or omits words that would be required to make it grammatical
1	The sentence is totally ungrammatical

Table 5: Annotators rated paraphrases along two 5-point scales.

8. **Averaged Substitution Constraints + LM** – As above, but including a language model probability.

4.4 Manual evaluation

We evaluated the paraphrase quality through a substitution test. We retrieved a number of sentences which contained each test phrase and substituted the phrase with automatically-generated paraphrases. Annotators judged whether the paraphrases had the same meaning as the original and whether the resulting sentences were grammatical. They assigned two values to each sentence using the 5-point scales given in Table 5. We considered an item to have the same meaning if it was assigned a score of 3 or greater, and to be grammatical if it was assigned a score of 4 or 5.

We evaluated several instances of a phrase when it occurred multiple times in the test corpus, since paraphrase quality can vary based on context (Szpektor et al., 2007). There were an average of 3.1 instances for each phrase, with a maximum of 6. There were a total of 1,195 sentences that paraphrases were substituted into, with a total of 8,422 judgements collected. Note that 7 different paraphrases were judged on average for every instance. This is because annotators judged paraphrases for eight conditions, and because we collected judgments for the 5-best paraphrases for many of the conditions.

We measured inter-annotator agreement with the Kappa statistic (Carletta, 1996) using the 1,391 items that two annotators scored in common. The two annotators assigned the same absolute score 47% of the time. If we consider chance agreement to be 20% for 5-point scales, then $K = 0.33$, which is commonly interpreted as "fair" (Landis and Koch, 1977). If we instead measure agreement in terms of how often the annotators both judged an item to be above or below the thresholds that we set, then their rate of agreement was 80%. In this case chance agreement would be 50%, so $K = 0.61$, which is "substantial".

4.5 Data and code

In order to allow other researchers to recreate our results or extend our work, we have prepared the following materials for download[2]:

- The complete set of paraphrases generated for the test set. This includes the 3.7 million paraphrases generated by the baseline method and the 3.5 million paraphrases generated with syntactic constraints.
- The code that we used to produce these paraphrases and the complete data sets (including all 10 word-aligned parallel corpora along with their English parses), so that researchers can extract paraphrases for new sets of phrases.
- The manual judgments about paraphrase quality. These may be useful as development material for setting the weights of a log-linear formulation of paraphrasing, as suggested in Zhao et al. (2008a).

5 Results

Table 6 summarizes the results of the manual evaluation. We can observe a strong trend in the syntactically constrained approaches performing better

[2]Available from `http://cs.jhu.edu/~ccb/`.

	correct meaning	correct grammar	both correct
Baseline	.56	.35	.30
Baseline+LM	.46	.44	.36
Extraction Constraints	**.62**	.57	.46
Extraction Const+LM	.60	.65	.50
Substitution Constraints	.60	.60	.50
Substitution Const+LM	.61	**.68**	.54
Avg Substitution Const	**.62**	.61	.51
Avg Substit Const+LM	.61	**.68**	**.55**

Table 6: The results of the manual evaluation for each of the eight conditions. Correct meaning is the percent of time that a condition was assigned a 3, 4, or 5, and correct grammar is the percent of time that it was given a 4 or 5, using the scales from Table 5.

than the baseline. They retain the correct meaning more often (ranging from 4% to up to 15%). They are judged to be grammatical far more frequently (up to 26% more often without the language model, and 24% with the language model) . They perform nearly 20% better when both meaning and grammaticality are used as criteria.[3]

Another trend that can be observed is that incorporating a language model probability tends to result in more grammatical output (a 7–9% increase), but meaning suffers as a result in some cases. When the LM is applied there is a drop of 12% in correct meaning for the baseline, but only a slight dip of 1-2% for the syntactically-constrained phrases.

Note that for the conditions where the paraphrases were required to have the same syntactic type as the phrase in the parse tree, there was a reduction in the number of paraphrases that could be applied. For the first two conditions, paraphrases were posited for 1194 sentences, conditions 3 and 4 could be applied to 1142 of those sentences, but conditions 5–8 could only be applied to 876 sentences. The substitution constraints reduce coverage to 73% of the test sentences. Given that the extraction constraints have better coverage and nearly identical performance on

[3]Our results show a significantly lower score for the baseline than reported in Bannard and Callison-Burch (2005). This is potentially due to the facts that in this work we evaluated on out-of-domain news commentary data, and we randomly selected phrases. In the pervious work the test phrases were drawn from WordNet, and they were evaluated solely on in-domain European parliament data.

the *meaning* criterion, they might be more suitable in some circumstances.

6 Conclusion

In this paper we have presented a novel refinement to paraphrasing with bilingual parallel corpora. We illustrated that a significantly higher performance can be achieved by constraining paraphrases to have the same syntactic type as the original phrase. A thorough manual evaluation found an absolute improvement in quality of 19% using strict criteria about paraphrase accuracy when comparing against a strong baseline. The syntactically enhanced paraphrases are judged to be grammatically correct over two thirds of the time, as opposed to the baseline method which was grammatically correct under half of the time.

This paper proposed constraints on paraphrases at two stages: when deriving them from parsed parallel corpora and when substituting them into parsed test sentences. These constraints produce paraphrases that are better than the baseline and which are less commonly affected by problems due to unaligned words. Furthermore, by introducing complex syntactic labels instead of solely relying on non-terminal symbols in the parse trees, we are able to keep the broad coverage of the baseline method.

Syntactic constraints significantly improve the quality of this paraphrasing method, and their use opens the question about whether analogous constraints can be usefully applied to paraphrases generated from purely monolingual corpora. Our improvements to the extraction of paraphrases from parallel corpora suggests that it may be usefully applied to other NLP applications, such as generation, which require grammatical output.

Acknowledgments

Thanks go to Sally Blatz, Emily Hinchcliff and Michelle Bland for conducting the manual evaluation and to Michelle Bland and Omar Zaidan for proofreading and commenting on a draft of this paper.

This work was supported by the National Science Foundation under Grant No. 0713448. The views and findings are the author's alone.

References

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of ACL*.

Regina Barzilay and Kathleen McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of ACL*.

Dan Bikel. 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. In *Proceedings of HLT*.

Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved statistical machine translation using paraphrases. In *Proceedings of HLT/NAACL*.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2007. (Meta-) evaluation of machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*.

Jean Carletta. 1996. Assessing agreement on classification tasks: The kappa statistic. *Computational Linguistics*, 22(2):249–254.

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

Heidi J. Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proceedings of EMNLP*.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT/NAACL*.

Bryant Huang and Kevin Knight. 2006. Relabeling syntax trees to improve syntax-based machine translation quality. In *Proceedings of HLT/NAACL*.

Ali Ibrahim, Boris Katz, and Jimmy Lin. 2003. Extracting structural paraphrases from aligned monolingual corpora. In *Proceedings of the Second International Workshop on Paraphrasing (ACL 2003)*.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for automatic evaluation. In *Proceedings of EMNLP*.

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

Philipp Koehn. 2005. A parallel corpus for statistical machine translation. In *Proceedings of MT-Summit*, Phuket, Thailand.

J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33:159–174.

Dekang Lin and Colin Cherry. 2002. Word alignment with cohesion constraint. In *Proceedings of HLT/NAACL*.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules from text. *Natural Language Engineering*, 7(3):343–360.

Nitin Madnani, Necip Fazil Ayan, Philip Resnik, and Bonnie Dorr. 2007. Using paraphrases for parameter tuning in statistical machine translation. In *Proceedings of the ACL Workshop on Statistical Machine Translation*.

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.

Karolina Owczarzak, Declan Groves, Josef Van Genabith, and Andy Way. 2006. Contextual bitext-derived paraphrases in automatic MT evaluation. In *Proceedings of the SMT Workshop at HLT-NAACL*.

Bo Pang, Kevin Knight, and Daniel Marcu. 2003. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proceedings of HLT/NAACL*.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal smt. In *Proceedings of ACL*.

Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. 2007. Statistical machine translation for query expansion in answer retrieval. In *Proceedings of ACL*.

Mark Steedman. 1999. Alternative quantier scope in ccg. In *Proceedings of ACL*.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, Denver, Colorado, September.

Idan Szpektor, Eyal Shnarch, and Ido Dagan. 2007. Instance-based evaluation of entailment rule acquisition. In *Proceedings of ACL*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3).

Shiqi Zhao, Cheng Niu, Ming Zhou, Ting Liu, and Sheng Li. 2008a. Combining multiple resources to improve SMT-based paraphrasing model. In *Proceedings of ACL/HLT*.

Shiqi Zhao, Haifeng Wang, Ting Liu, and Sheng Li. 2008b. Pivot approach for extracting paraphrase patterns from bilingual corpora. In *Proceedings of ACL/HLT*.

Liang Zhou, Chin-Yew Lin, and Eduard Hovy. 2006. Reevaluating machine translation results with paraphrase support. In *Proceedings of EMNLP*.

Forest-based Translation Rule Extraction

Haitao Mi[1]
[1]Key Lab. of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
P.O. Box 2704, Beijing 100190, China
htmi@ict.ac.cn

Liang Huang[2,1]
[2]Dept. of Computer & Information Science
University of Pennsylvania
3330 Walnut St., Levine Hall
Philadelphia, PA 19104, USA
lhuang3@cis.upenn.edu

Abstract

Translation rule extraction is a fundamental problem in machine translation, especially for *linguistically syntax-based* systems that need parse trees from either or both sides of the bitext. The current dominant practice only uses 1-best trees, which adversely affects the rule set quality due to parsing errors. So we propose a novel approach which extracts rules from a *packed forest* that compactly encodes exponentially many parses. Experiments show that this method improves translation quality by over 1 BLEU point on a state-of-the-art tree-to-string system, and is 0.5 points better than (and twice as fast as) extracting on 30-best parses. When combined with our previous work on forest-based decoding, it achieves a 2.5 BLEU points improvement over the baseline, and even outperforms the hierarchical system of Hiero by 0.7 points.

1 Introduction

Automatic extraction of translation rules is a fundamental problem in statistical machine translation, especially for many syntax-based models where translation rules directly encode linguistic knowledge. Typically, these models extract rules using parse trees from *both* or *either* side(s) of the bitext. The former case, with trees on both sides, is often called *tree-to-tree* models; while the latter case, with trees on either source or target side, include both *tree-to-string* and *string-to-tree* models (see Table 1). Leveraging from structural and linguistic information from parse trees, these models are believed to be better than their phrase-based counterparts in handling non-local reorderings, and have achieved promising translation results.[1]

source target	*examples (partial)*
tree-to-tree	Ding and Palmer (2005)
tree-to-string	Liu et al. (2006); Huang et al. (2006)
string-to-tree	Galley et al. (2006)
string-to-string	Chiang (2005)

Table 1: A classification of syntax-based MT. The first three use *linguistic syntax*, while the last one only *formal syntax*. Our experiments cover the second type using a packed forest in place of the tree for rule-extraction.

However, these systems suffer from a major limitation, that the rule extractor only uses 1-best parse tree(s), which adversely affects the rule set quality due to parsing errors. To make things worse, modern statistical parsers are often trained on domains quite different from those used in MT. By contrast, *formally syntax-based* models (Chiang, 2005) do not rely on parse trees, yet usually perform better than these linguistically sophisticated counterparts.

To alleviate this problem, an obvious idea is to extract rules from k-best parses instead. However, a k-best list, with its limited scope, has too few variations and too many redundancies (Huang, 2008). This situation worsens with longer sentences as the number of possible parses grows exponentially with the sentence length and a k-best list will only capture a tiny fraction of the whole space. In addition, many subtrees are repeated across different parses, so it is

[1]For example, in recent NIST Evaluations, some of these models (Galley et al., 2006; Quirk et al., 2005; Liu et al., 2006) ranked among top 10. See http://www.nist.gov/speech/tests/mt/.

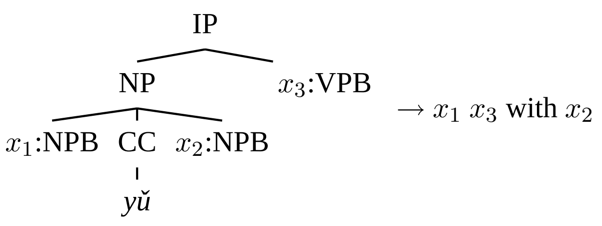


Figure 1: Example translation rule r_1. The Chinese conjunction *yǔ* "and" is translated into English prep. "with".

also inefficient to extract rules separately from each of these very similar trees (or from the cross-product of k^2 similar tree-pairs in tree-to-tree models).

We instead propose a novel approach that extracts rules from *packed forests* (Section 3), which compactly encodes many more alternatives than k-best lists. Experiments (Section 5) show that forest-based extraction improves BLEU score by over 1 point on a state-of-the-art tree-to-string system (Liu et al., 2006; Mi et al., 2008), which is also 0.5 points better than (and twice as fast as) extracting on 30-best parses. When combined with our previous orthogonal work on forest-based decoding (Mi et al., 2008), the forest-forest approach achieves a 2.5 BLEU points improvement over the baseline, and even outperforms the hierarchical system of Hiero, one of the best-performing systems to date.

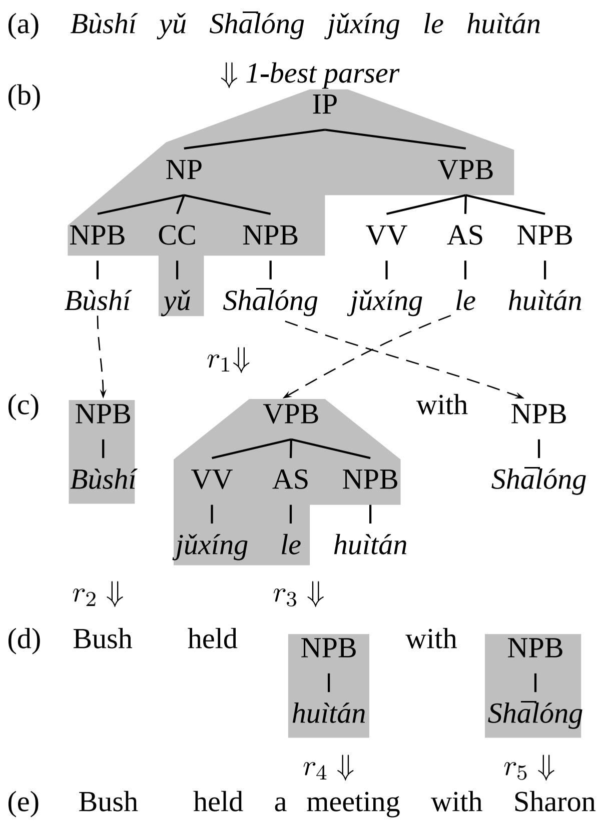
Besides tree-to-string systems, our method is also applicable to other paradigms such as the string-to-tree models (Galley et al., 2006) where the rules are in the reverse order, and easily generalizable to pairs of forests in tree-to-tree models.

2 Tree-based Translation

We review in this section the tree-based approach to machine translation (Liu et al., 2006; Huang et al., 2006), and its rule extraction algorithm (Galley et al., 2004; Galley et al., 2006).

2.1 Tree-to-String System

Current tree-based systems perform translation in two separate steps: parsing and decoding. The input string is first parsed by a parser into a 1-best tree, which will then be converted to a target language string by applying a set of tree-to-string transformation rules. For example, consider the following example translating from Chinese to English:



r_2	NPB(*Bùshí*) → Bush
r_3	VPB(VV(*jǔxíng*) AS(*le*) x_1:NPB) → held x_1
r_4	NPB(*Shālóng*) → Sharon
r_5	NPB(*huìtán*) → a meeting

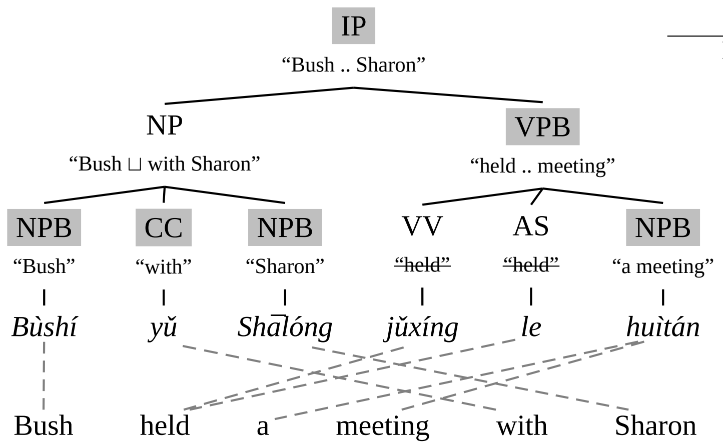
Figure 2: Example derivation of tree-to-string translation, with rules used. Each shaded region denotes a tree fragment that is pattern-matched with the rule being applied.

(1)	*Bùshí*	*yǔ*	*Shālóng*	*jǔxíng*	*le*	*huìtán*
	Bush	and/with	Sharon$_1$	hold	*past.*	meeting$_2$

"Bush held a meeting$_2$ with Sharon$_1$"

Figure 2 shows how this process works. The Chinese sentence (a) is first parsed into a parse tree (b), which will be converted into an English string in 5 steps. First, at the root node, we apply rule r_1 shown in Figure 1, which translates the Chinese coordination construction ("... and ...") into an English prepositional phrase. Then, from step (c) we continue applying rules to untranslated Chinese subtrees, until we get the complete English translation in (e).[2]

[2]We swap the 1-best and 2-best parses of the example sentence from our earlier paper (Mi et al., 2008), since the current 1-best parse is easier to illustrate the rule extraction algorithm.



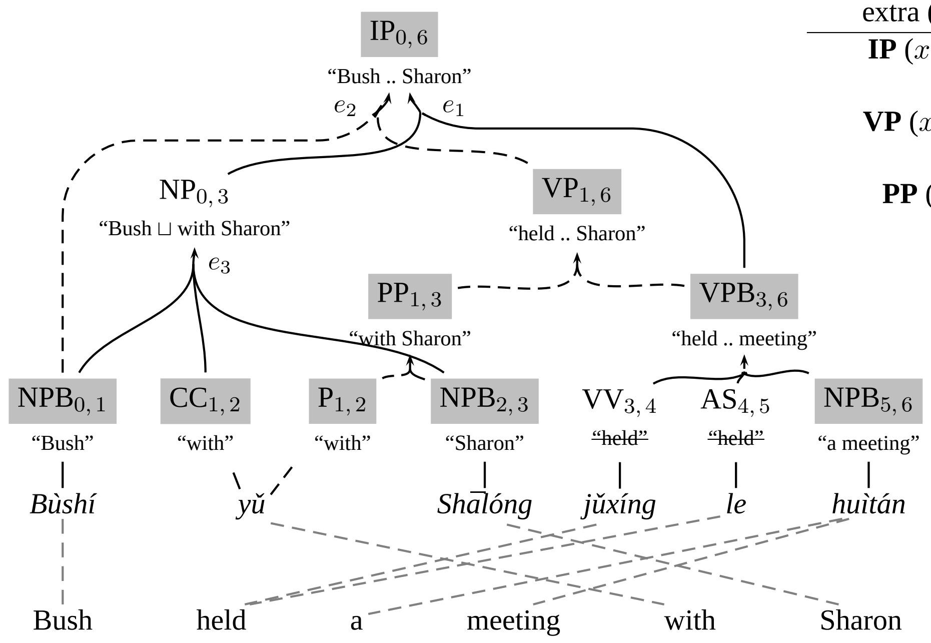
(minimal) rules extracted

IP (NP(x_1:NPB x_2:CC x_3:NPB) x_4:VPB)
$\rightarrow x_1\ x_4\ x_2\ x_3$

CC (*yǔ*) → with
NPB (*Bùshí*) → Bush
NPB (*Shālóng*) → Sharon

VPB (VV(*jǔxíng*) AS(*le*) x_1:NPB)
→ held x_1
NPB (*huìtán*) → a meeting

Figure 3: Tree-based rule extraction (Galley et al., 2004). Each non-leaf node in the tree is annotated with its target span (below the node), where ⊔ denotes a gap, and non-faithful spans are crossed out. Shadowed nodes are **admissible**, with contiguous and faithful spans. The first two rules can be "composed" to form rule r_1 in Figure 1.



extra (minimal) rules extracted

IP (x_1:NPB x_2:VP) → $x_1\ x_2$

VP (x_1:PP x_2:VPB) → $x_2\ x_1$

PP (x_1:P x_2:NPB) → $x_1\ x_2$

P (*yǔ*) → with

Figure 4: Forest-based rule extraction. Solid hyperedges correspond to the 1-best tree in Figure 3, while dashed hyperedges denote the alternative parse interpreting *yǔ* as a preposition in Figure 5.

More formally, a (tree-to-string) **translation rule** (Galley et al., 2004; Huang et al., 2006) is a tuple $\langle lhs(r), rhs(r), \phi(r)\rangle$, where $lhs(r)$ is the source-side tree fragment, whose internal nodes are labeled by nonterminal symbols (like NP and VP), and whose frontier nodes are labeled by source-language words (like "*yǔ*") or variables from a set $\mathcal{X} = \{x_1, x_2, \ldots\}$; $rhs(r)$ is the target-side string expressed in target-language words (like "with") and variables; and $\phi(r)$ is a mapping from $\mathcal{X}$ to nonterminals. Each variable $x_i \in \mathcal{X}$ occurs *exactly once* in $lhs(r)$ and *exactly once* in $rhs(r)$. For example, for rule r_1 in Figure 1,

$$\begin{aligned} lhs(r_1) &= \text{IP (NP(}x_1 \text{ CC(}yǔ\text{) } x_2\text{)} \ \ x_3\text{)},\\ rhs(r_1) &= x_1 \ x_3 \text{ with } x_2,\\ \phi(r_1) &= \{x_1\text{: NPB, } x_2\text{: NPB, } x_3\text{: VPB}\}. \end{aligned}$$

These rules are being used in the reverse direction of the string-to-tree transducers in Galley et al. (2004).

2.2 Tree-to-String Rule Extraction

We now briefly explain the algorithm of Galley et al. (2004) that can extract these translation rules from a word-aligned bitext with source-side parses.

Consider the example in Figure 3. The basic idea is to decompose the source (Chinese) parse into a series of tree fragments, each of which will form a rule with its corresponding English translation. However, not every fragmentation can be used for rule extraction, since it may or may not respect the alignment and reordering between the two languages. So we say a fragmentation is **well-formed** with respect to an alignment if the root node of every tree fragment corresponds to a **contiguous span** on the target side; the intuition is that there is a "translational equivalence" between the subtree rooted at the node and the corresponding target span. For example, in Figure 3, each node is annotated with its corresponding English span, where the NP node maps to a non-contiguous one "Bush $\sqcup$ with Sharon".

More formally, we need a precise formulation to handle the cases of one-to-many, many-to-one, and many-to-many alignment links. Given a source-target sentence pair (σ, τ) with alignment a, the (target) **span** of node v is the set of target words aligned to leaf nodes $yield(v)$ under node v:

$$span(v) \triangleq \{\tau_i \in \tau \mid \exists \sigma_j \in yield(v), (\sigma_j, \tau_i) \in a\}.$$

For example, in Figure 3, every node in the parse tree is annotated with its corresponding span below the node, where most nodes have contiguous spans except for the NP node which maps to a gapped phrase "Bush $\sqcup$ with Sharon". But contiguity alone is not enough to ensure well-formedness, since there might be words within the span aligned to source words uncovered by the node. So we also define a span s to be **faithful** to node v if every word in it is *only* aligned to nodes dominated by v, i.e.:

$$\forall \tau_i \in s, (\sigma_j, \tau_i) \in a \Rightarrow \sigma_j \in yield(v).$$

For example, sibling nodes VV and AS in the tree have non-faithful spans (crossed out in the Figure), because they both map to "held", thus *neither* of them can be translated to "held" alone. In this case, a larger tree fragment rooted at VPB has to be extracted. Nodes with non-empty, contiguous, *and* faithful spans form the **admissible set** (shaded nodes in the figure), which serve as potential cut-points for rule extraction.[3]

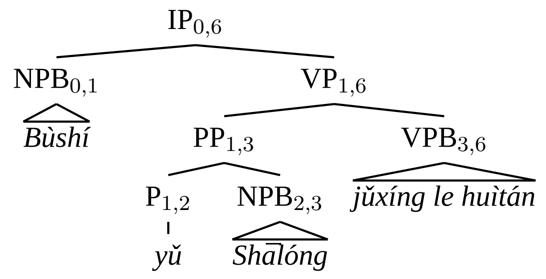


Figure 5: An alternative parse of the Chinese sentence, with *yǔ* as a preposition instead of a conjunction; common parts shared with 1-best parse in Fig. 3 are elided.

With the admissible set computed, rule extraction is as simple as a depth-first traversal from the root: we "cut" the tree at all admissible nodes to form tree fragments and extract a rule for each fragment, with variables matching the admissible descendant nodes. For example, the tree in Figure 3 is cut into 6 pieces, each of which corresponds to a rule on the right.

These extracted rules are called *minimal rules*, which can be glued together to form *composed rules* with larger tree fragments (e.g. r_1 in Fig. 1) (Galley et al., 2006). Our experiments use composed rules.

3 Forest-based Rule Extraction

We now extend tree-based extraction algorithm from the previous section to work with a packed forest representing exponentially many parse trees.

3.1 Packed Forest

Informally, a packed parse forest, or *forest* in short, is a compact representation of all the derivations (i.e., parse trees) for a given sentence under a context-free grammar (Earley, 1970; Billot and Lang, 1989). For example, consider again the Chinese sentence in Example (1) above, which has (at least) two readings depending on the part-of-speech of the word *yǔ*: it can be either a conjunction (CC "and") as shown in Figure 3, or a preposition (P "with") as shown in Figure 5, with only PP and VPB swapped from the English word order.

[3]Admissible set (Wang et al., 2007) is also known as "frontier set" (Galley et al., 2004). For simplicity of presentation, we assume every target word is aligned to at least one source word; see Galley et al. (2006) for handling unaligned target words.

These two parse trees can be represented as a single forest by sharing common subtrees such as $\text{NPB}_{0,1}$ and $\text{VPB}_{3,6}$, as shown in Figure 4. Such a forest has a structure of a *hypergraph* (Huang and Chiang, 2005), where items like $\text{NP}_{0,3}$ are called *nodes*, whose indices denote the source span, and combinations like

$$e_1 : \text{IP}_{0,6} \rightarrow \text{NPB}_{0,3}\ \text{VP}_{3,6}$$

we call *hyperedges*. We denote $head(e)$ and $tails(e)$ to be the consequent and antecedant items of hyperedge e, respectively. For example,

$$head(e_1) = \text{IP}_{0,6},\ \ tails(e_1) = \{\text{NPB}_{0,3}, \text{VP}_{3,6}\}.$$

We also denote $BS(v)$ to be the set of **incoming hyperedges** of node v, being different ways of deriving it. For example, in Figure 4, $BS(\text{IP}_{0,6}) = \{e_1, e_2\}$.

3.2 Forest-based Rule Extraction Algorithm

Like in tree-based extraction, we extract rules from a packed forest F in two steps:

(1) admissible set computation (where to cut), and

(2) fragmentation (how to cut).

It turns out that the exact formulation developed for admissible set in the tree-based case can be applied to a forest without any change. The fragmentation step, however, becomes much more involved since we now face a choice of multiple parse hyperedges at each node. In other words, it becomes *non-deterministic* how to "cut" a forest into tree fragments, which is analogous to the non-deterministic pattern-match in forest-based decoding (Mi et al., 2008). For example there are two parse hyperedges e_1 and e_2 at the root node in Figure 4. When we follow one of them to grow a fragment, there again will be multiple choices at each of its tail nodes. Like in tree-based case, a fragment is said to be **complete** if all its leaf nodes are admissible. Otherwise, an incomplete fragment can grow at any non-admissible frontier node v, where following each parse hyperedge at v will split off a new fragment. For example, following e_2 at the root node will immediately lead us to two admissible nodes, $\text{NPB}_{0,1}$ and $\text{VP}_{1,6}$ (we will highlight admissible nodes by gray shades in this section like in Figures 3 and 4). So this fragment, $frag_1 = \{e_2\}$, is now complete and we can extract a rule,

$$\textbf{IP}\ (x_1\text{:NPB}\ \ x_2\text{:VP}) \rightarrow x_1\ x_2.$$

However, following the other hyperedge e_1

$$\text{IP}_{0,6} \rightarrow \text{NP}_{0,3}\ \text{VPB}_{3,6}$$

will leave the new fragment $frag_2 = \{e_1\}$ *incomplete* with one non-admissible node $\text{NP}_{0,3}$. We then grow $frag_2$ at this node by choosing hyperedge e_3

$$\text{NP}_{0,3} \rightarrow \text{NPB}_{0,1}\ \text{CC}_{1,2}\ \text{NPB}_{2,3},$$

and spin off a new fragment $frag_3 = \{e_1, e_3\}$, which is now complete since all its four leaf nodes are admissible. We then extract a rule with four variables:

$$\textbf{IP}\ (\text{NP}(x_1\text{:NPB}\ x_2\text{:CC}\ x_3\text{:NPB})\ x_4\text{:VPB}) \rightarrow\ x_1\, x_4\, x_2\, x_3.$$

This procedure is formalized by a breadth-first search (BFS) in Pseudocode 1. The basic idea is to visit each frontier node v, and keep a queue *open* of actively growing fragments rooted at v. We keep expanding incomplete fragments from *open*, and extract a rule if a complete fragment is found (line 10). Each fragment is associated with a *frontier* (variable

Algorithm 1 Forest-based Rule Extraction.

```
    Input: forest $F$, target sentence $\tau$, and alignment $a$
    Output: minimal rule set $\mathcal{R}$
 1: $admset \leftarrow \textsc{Admissible}(F, \tau, a)$        ▷ admissible set
 2: for each $v \in admset$ do
 3:     $open \leftarrow \varnothing$                    ▷ queue of active fragments
 4:     for each $e \in BS(v)$ do          ▷ incoming hyperedges
 5:         $front \leftarrow tails(e) \setminus admset$       ▷ initial frontier
 6:         $open$.append($\langle\{e\}, front\rangle$)
 7:     while $open \neq \varnothing$ do
 8:         $\langle frag, front\rangle \leftarrow open$.pop()     ▷ active fragment
 9:         if $front = \varnothing$ then
10:             generate a rule $r$ using fragment $frag$
11:             $\mathcal{R}$.append($r$)
12:         else                    ▷ incomplete: further expand
13:             $u \leftarrow front$.pop()                ▷ a frontier node
14:             for each $e \in BS(u)$ do
15:                 $front' \leftarrow front \cup (tails(e) \setminus admset)$
16:                 $open$.append($\langle frag \cup \{e\}, front'\rangle$)
```

front in the Pseudocode), being the subset of non-admissible leaf nodes (recall that expansion stops at admissible nodes). So each initial fragment along hyperedge e is associated with an initial frontier (line 5), $\mathit{front} = \mathit{tails}(e) \setminus \mathit{admset}$.

A fragment is complete if its frontier is empty (line 9), otherwise we pop one frontier node u to expand, spin off new fragments by following hyperedges of u, and update the frontier (lines 14-16), until all active fragments are complete and *open* queue is empty (line 7).

A single parse tree can also be viewed as a trivial forest, where each node has only one incoming hyperedge. So the Galley et al. (2004) algorithm for tree-based rule extraction (Sec. 2.2) can be considered a special case of our algorithm, where the queue *open* always contains one single active fragment.

3.3 Fractional Counts and Rule Probabilities

In tree-based extraction, for each sentence pair, each rule extracted naturally has a count of one, which will be used in maximum-likelihood estimation of rule probabilities. However, a forest is an implicit collection of many more trees, each of which, when enumerated, has its own probability accumulated from of the parse hyperedges involved. In other words, a forest can be viewed as a virtual weighted k-best list with a huge k. So a rule extracted from a non 1-best parse, i.e., using non 1-best hyperedges, should be penalized accordingly and should have a *fractional count* instead of a unit one, similar to the E-step in EM algorithms.

Inspired by the parsing literature on pruning (Charniak and Johnson, 2005; Huang, 2008) we penalize a rule r by the posterior probability of its tree fragment $\mathit{frag} = \mathit{lhs}(r)$. This posterior probability, notated $\alpha\beta(\mathit{frag})$, can be computed in an Inside-Outside fashion as the product of three parts: the outside probability of its root node, the probabilities of parse hyperedges involved in the fragment, and the inside probabilities of its leaf nodes,

$$\alpha\beta(\mathit{frag}) = \alpha(\mathit{root}(\mathit{frag})) \cdot \prod_{e \in \mathit{frag}} \mathrm{P}(e) \cdot \prod_{v \in \mathit{yield}(\mathit{frag})} \beta(v) \quad (2)$$

where $\alpha(\cdot)$ and $\beta(\cdot)$ denote the outside and inside probabilities of tree nodes, respectively. For example in Figure 4,

$$\alpha\beta(\{e_2, e_3\}) = \alpha(\mathrm{IP}_{0,6}) \cdot \mathrm{P}(e_2) \cdot \mathrm{P}(e_3) \cdot \beta(\mathrm{NPB}_{0,1})\beta(\mathrm{CC}_{1,2})\beta(\mathrm{NPB}_{2,3})\beta(\mathrm{VPB}_{3,6}).$$

Now the fractional count of rule r is simply

$$c(r) = \frac{\alpha\beta(\mathit{lhs}(r))}{\alpha\beta(\mathrm{TOP})} \quad (3)$$

where TOP denotes the root node of the forest.

Like in the M-step in EM algorithm, we now extend the maximum likelihood estimation to fractional counts for three conditional probabilities regarding a rule, which will be used in the experiments:

$$\mathrm{P}(r \mid \mathit{lhs}(r)) = \frac{c(r)}{\sum_{r':\mathit{lhs}(r')=\mathit{lhs}(r)} c(r')}, \quad (4)$$

$$\mathrm{P}(r \mid \mathit{rhs}(r)) = \frac{c(r)}{\sum_{r':\mathit{rhs}(r')=\mathit{rhs}(r)} c(r')}, \quad (5)$$

$$\mathrm{P}(r \mid \mathit{root}(\mathit{lhs}(r))) = \frac{c(r)}{\sum_{r':\mathit{root}(\mathit{lhs}(r'))=\mathit{root}(\mathit{lhs}(r))} c(r')}. \quad (6)$$

4 Related Work

The concept of *packed forest* has been previously used in translation rule extraction, for example in rule composition (Galley et al., 2006) and tree binarization (Wang et al., 2007). However, both of these efforts only use 1-best parses, with the second one packing different *binarizations* of the same tree in a forest. Nevertheless we suspect that their extraction algorithm is in principle similar to ours, although they do not provide details of forest-based fragmentation (Algorithm 1) which we think is non-trivial.

The forest concept is also used in machine translation decoding, for example to characterize the search space of decoding with integrated language models (Huang and Chiang, 2007). The first *direct* application of parse forest in translation is our previous work (Mi et al., 2008) which translates a packed forest from a parser; it is also the base system in our experiments (see below). This work, on the other hand, is in the orthogonal direction, where we utilize forests in rule extraction instead of decoding.

Our experiments will use both default 1-best decoding and forest-based decoding. As we will see in the next section, the best result comes when we combine the merits of both, i.e., using forests in both rule extraction and decoding.

There is also a parallel work on extracting rules from k-best parses and k-best alignments (Venugopal et al., 2008), but both their experiments and our own below confirm that extraction on k-best parses is neither efficient nor effective.

5 Experiments

5.1 System

Our experiments are on Chinese-to-English translation based on a tree-to-string system similar to (Huang et al., 2006; Liu et al., 2006). Given a 1-best tree T, the decoder searches for the best derivation d^* among the set of all possible derivations D:

$$d^* = \arg\max_{d \in D} \lambda_0 \log \mathrm{P}(d \mid T) + \lambda_1 \log \mathrm{P}_{\mathrm{lm}}(\tau(d)) + \lambda_2 |d| + \lambda_3 |\tau(d)| \tag{7}$$

where the first two terms are translation and language model probabilities, $\tau(d)$ is the target string (English sentence) for derivation d, and the last two terms are derivation and translation length penalties, respectively. The conditional probability $\mathrm{P}(d \mid T)$ decomposes into the product of rule probabilities:

$$\mathrm{P}(d \mid T) = \prod_{r \in d} \mathrm{P}(r). \tag{8}$$

Each $\mathrm{P}(r)$ is in turn a product of five probabilities:

$$\begin{aligned} \mathrm{P}(r) &= \mathrm{P}(r \mid \mathit{lhs}(r))^{\lambda_4} \cdot \mathrm{P}(r \mid \mathit{rhs}(r))^{\lambda_5} \\ &\cdot \mathrm{P}(r \mid \mathit{root}(\mathit{lhs}(r)))^{\lambda_6} \\ &\cdot \mathrm{P}_{\mathrm{lex}}(\mathit{lhs}(r) \mid \mathit{rhs}(r))^{\lambda_7} \\ &\cdot \mathrm{P}_{\mathrm{lex}}(\mathit{rhs}(r) \mid \mathit{lhs}(r))^{\lambda_8} \end{aligned} \tag{9}$$

where the first three are conditional probabilities based on fractional counts of rules defined in Section 3.3, and the last two are lexical probabilities. These parameters $\lambda_1 \ldots \lambda_8$ are tuned by minimum error rate training (Och, 2003) on the dev sets. We refer readers to Mi et al. (2008) for details of the decoding algorithm.

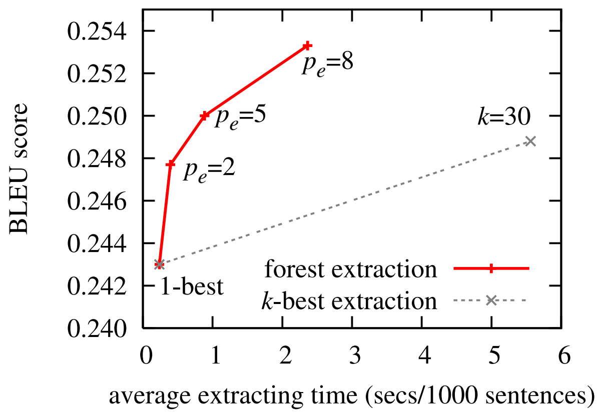


Figure 6: Comparison of extraction time and BLEU score: forest-based vs.1-best and 30-best.

rules from...	extraction	decoding	BLEU
1-best trees	0.24	1.74	0.2430
30-best trees	5.56	3.31	0.2488
forest: p_e=8	2.36	3.40	**0.2533**
Pharaoh	-	-	0.2297

Table 2: Results with different rule extraction methods. Extraction and decoding columns are running times in secs per 1000 sentences and per sentence, respectively.

We use the Chinese parser of Xiong et al. (2005) to parse the source side of the bitext. Following Huang (2008), we also modify this parser to output a packed forest for each sentence, which can be pruned by the marginal probability-based inside-outside algorithm (Charniak and Johnson, 2005; Huang, 2008). We will first report results trained on a small-scaled dataset with detailed analysis, and then scale to a larger one, where we also combine the technique of forest-based decoding (Mi et al., 2008).

5.2 Results and Analysis on Small Data

To test the effect of forest-based rule extraction, we parse the training set into parse forests and use three levels of pruning thresholds: $p_e = 2, 5, 8$.

Figure 6 plots the extraction speed and translation quality of forest-based extraction with various pruning thresholds, compared to 1-best and 30-best baselines. Using more than one parse tree apparently improves the BLEU score, but at the cost of much slower extraction, since each of the top-k trees has to be processed individually although they share many

rules from ...	total #	on dev	new rules used
1-best trees	440k	90k	-
30-best trees	1.2M	130k	8.71%
forest: p_e=8	3.3M	188k	16.3%

Table 3: Statistics of rules extracted from small data. The last column shows the ratio of *new* rules introduced by non 1-best parses being used in 1-best derivations.

extract. \ decoding	1-best tree	forest: p_d=10
1-best trees	0.2560	0.2674
30-best trees	0.2634	0.2767
forest: p_e=5	0.2679	**0.2816**
Hiero	0.2738	

Table 4: BLEU score results trained on large data.

common subtrees. Forest extraction, by contrast, is much faster thanks to packing and produces consistently better BLEU scores. With pruning threshold $p_e = 8$, forest-based extraction achieves a (case insensitive) BLEU score of 0.2533, which is an absolute improvement of 1.0% points over the 1-best baseline, and is statistically significant using the *sign-test* of Collins et al. (2005) ($p < 0.01$). This is also 0.5 points better than (and twice as fast as) extracting on 30-best parses. These BLEU score results are summarized in Table 2, which also shows that decoding with forest-extracted rules is less than twice as slow as with 1-best rules, and only fractionally slower than with 30-best rules.

We also investigate the question of how often rules extracted from non 1-best parses are used by the decoder. Table 3 shows the numbers of rules extracted from 1-best, 30-best and forest-based extractions, and the numbers that survive after filtering on the dev set. Basically in the forest-based case we can use about twice as many rules as in the 1-best case, or about 1.5 times of 30-best extraction. But the real question is, are these extra rules really useful in generating the final (1-best) translation? The last row shows that 16.3% of the rules used in 1-best derivations are indeed *only* extracted from non 1-best parses in the forests. Note that this is a stronger condition than changing the distribution of rules by considering more parses; here we introduce *new* rules never seen on any 1-best parses.

5.3 Final Results on Large Data

We also conduct experiments on a larger training dataset, FBIS, which contains 239K sentence pairs with about 6.9M/8.9M words in Chinese/English, respectively. We also use a bigger trigram model trained on the first 1/3 of the Xinhua portion of Gigaword corpus. To integrate with forest-based decoding, we use both 1-best trees and packed forests during both rule extraction and decoding phases. Since the data scale is larger than the small data, we are forced to use harsher pruning thresholds, with $p_e = 5$ for extraction and $p_d = 10$ for decoding.

The final BLEU score results are shown in Table 4. With both tree-based and forest-based decoding, rules extracted from forests significantly outperform those extracted from 1-best trees ($p < 0.01$). The final result with both forest-based extraction and forest-based decoding reaches a BLEU score of 0.2816, outperforming that of Hiero (Chiang, 2005), one of the best performing systems to date. These results confirm that our novel forest-based rule extraction approach is a promising direction for syntax-based machine translation.

6 Conclusion and Future Work

In this paper, we have presented a novel approach that extracts translation rules from a packed forest encoding exponentially many trees, rather than from 1-best or k-best parses. Experiments on a state-of-the-art tree-to-string system show that this method improves BLEU score significantly, with reasonable extraction speed. When combined with our previous work on forest-based decoding, the final result is even better than the hierarchical system Hiero. For future work we would like to apply this approach to other types of syntax-based translation systems, namely the string-to-tree systems (Galley et al., 2006) and tree-to-tree systems.

Acknowledgement

This work was funded by National Natural Science Foundation of China, Contracts 60736014 and 60573188, and 863 State Key Project No. 2006AA010108 (H. M.), and by NSF ITR EIA-0205456 (L. H.). We would also like to thank Qun Liu for supporting this work, and the three anonymous reviewers for improving the earlier version.

References

Sylvie Billot and Bernard Lang. 1989. The structure of shared forests in ambiguous parsing. In *Proceedings of ACL '89*, pages 143–151.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine-grained n-best parsing and discriminative reranking. In *Proceedings of the 43rd ACL*, Ann Arbor, MI.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd ACL*, Ann Arbor, MI.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL*, pages 531–540, Ann Arbor, Michigan, June.

Yuan Ding and Martha Palmer. 2005. Machine translation using probablisitic synchronous dependency insertion grammars. In *Proceedings of the 43rd ACL*, Ann Arbor, MI.

Jay Earley. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT-NAACL*, pages 273–280.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of COLING-ACL*.

Liang Huang and David Chiang. 2005. Better k-best Parsing. In *Proceedings of the Ninth International Workshop on Parsing Technologies (IWPT-2005)*.

Liang Huang and David Chiang. 2007. Forest rescoring: Fast decoding with integrated language models. In *Proceedings of ACL*, Prague, Czech Rep., June.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of AMTA*, Boston, MA, August.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of the ACL: HLT*, Columbus, OH, June.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of COLING-ACL*, pages 609–616.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of ACL: HLT*, Columbus, OH.

Franz Joseph Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal smt. In *Proceedings of the 43rd ACL*, Ann Arbor, MI.

Ashish Venugopal, Andreas Zollmann, Noah A. Smith, and Stephan Vogel. 2008. Wider pipelines: N-best alignments and parses in mt training. In *Proceedings of AMTA*, Honolulu, Hawaii.

Wei Wang, Kevin Knight, and Daniel Marcu. 2007. Binarizing syntax trees to improve syntax-based machine translation accuracy. In *Proceedings of EMNLP*, Prague, Czech Rep., July.

Deyi Xiong, Shuanglong Li, Qun Liu, and Shouxun Lin. 2005. Parsing the penn chinese treebank with semantic knowledge. In *Proceedings of IJCNLP 2005*, pages 70–81.

Probabilistic Inference for Machine Translation

Phil Blunsom and **Miles Osborne**
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh, EH8 9AB, UK
{pblunsom,miles}@inf.ed.ac.uk

Abstract

We advance the state-of-the-art for discriminatively trained machine translation systems by presenting novel probabilistic inference and search methods for synchronous grammars. By approximating the intractable space of all candidate translations produced by intersecting an ngram language model with a synchronous grammar, we are able to train and decode models incorporating millions of sparse, heterogeneous features. Further, we demonstrate the power of the discriminative training paradigm by extracting structured syntactic features, and achieving increases in translation performance.

1 Introduction

The goal of creating statistical machine translation (SMT) systems incorporating rich, sparse, features over syntax and morphology has consumed much recent research attention. Discriminative approaches are widely seen as a promising technique, potentially allowing us to further the state-of-the-art. Most work on discriminative training for SMT has focussed on linear models, often with margin based algorithms (Liang et al., 2006; Watanabe et al., 2006), or rescaling a product of sub-models (Och, 2003; Ittycheriah and Roukos, 2007).

Recent work by Blunsom et al. (2008) has shown how translation can be framed as a probabilistic log-linear model, where the distribution over translations is modelled in terms of a latent variable on derivations. Their approach was globally optimised and discriminative trained. However, a language model, an information source known to be crucial for obtaining good performance in SMT, was notably omitted. This was because adding a language model would mean that the normalising partition function could no longer be exactly calculated, thereby preventing efficient parameter estimation.

Here, we show how language models can be incorporated into large-scale discriminative translation models, without losing the probabilistic interpretation of the model. The key insight is that we can use Monte-Carlo methods to approximate the partition function, thereby allowing us to tackle the extra computational burden associated with adding the language model. This approach is theoretically justified and means that the model continues to be both probabilistic and globally optimised. As expected, using a language model dramatically increases translation performance.

Our second major contribution is an exploitation of syntactic features. By encoding source syntax as features allows the model to use, or ignore, this information as it sees fit, thereby avoiding the problems of coverage and sparsity associated with directly incorporating the syntax into the grammar (Huang et al., 2006; Mi et al., 2008). We report on translation gains using this approach.

We begin by introducing the synchronous grammar approach to SMT in Section 2. In Section 3 we define the parametric form of our model and describe techniques for approximating the intractable space of all translations for a given source sentence. In Section 4 we evaluate the ability of our model to effectively estimate the highly dependent weights for the sparse features and real-valued language model. In addition we describe how

S ⇒ ⟨X_1 的 。, X_1 .⟩

X_1 ⇒ ⟨X_2 是 昨天 深夜 X_3 , X_2 X_3 late last night⟩

X_2 ⇒ ⟨布朗 , Brown⟩

X_3 ⇒ ⟨从 X_4 抵达 X_5 , arrived in X_5 from X_4⟩

X_4 ⇒ ⟨上海 , Shanghai⟩

X_5 ⇒ ⟨北京 , Beijing⟩

Figure 1. An example SCFG derivation from a Chinese source sentence which yields the English sentence: *"Brown arrived in Shanghai from Beijing late last night."*

our model can easily integrate rich features over source syntax trees and compare our training methods to a state-of-the-art benchmark.

2 Synchronous context free grammar

A synchronous context free grammar (SCFG, (Lewis II and Stearns, 1968)) describes the generation of pairs of strings. A string pair is generated by applying a series of paired context-free rewrite rules of the form, $X \rightarrow \langle \alpha, \gamma, \sim \rangle$, where X is a non-terminal, α and γ are strings of terminals and non-terminals and $\sim$ specifies a one-to-one alignment between non-terminals in α and γ. In the context of SMT, by assigning the source and target languages to the respective sides of a SCFG it is possible to describe translation as the process of parsing the source sentence, while generating the target translation (Chiang, 2007).

In this paper we only consider grammars extracted using the heuristics described for the Hiero SMT system (Chiang, 2007). Note however that our approach is general and could be used with other synchronous grammar transducers (e.g., (Galley et al., 2006)). SCFG productions can specify that the order of the child non-terminals is the same in both languages (a *monotone* production), or is reversed (a *reordering* production). Without loss of generality, here we add the restriction that non-terminals on the source and target sides of the grammar must have the same category. Figure 1 shows an example derivation for Chinese to English translation.

3 Model

We start by defining a log-linear model for the conditional probability distribution over target translations of a given source sentence. A sequence of SCFG rule applications which produce a translation from a source sentence is referred to as a *derivation*, and each translation may be produced by many different derivations. As the training data only provides source and target sentences, the derivations are modelled as a latent variable.

The conditional probability of a derivation, $\mathbf{d}$, for a target translation, $\mathbf{e}$, conditioned on the source, $\mathbf{f}$, is given by:

$$p_\Lambda(\mathbf{d}, \mathbf{e}|\mathbf{f}) = \frac{\exp \sum_k \lambda_k H_k(\mathbf{d}, \mathbf{e}, \mathbf{f})}{Z_\Lambda(\mathbf{f})} \quad (1)$$

$$\text{where} \quad H_k(\mathbf{d}, \mathbf{e}, \mathbf{f}) = \sum_{r \in \mathbf{d}} h_k(\mathbf{f}, r, q(r, \mathbf{d})) \quad (2)$$

Using Equation (1), the conditional probability of a target translation given the source is the sum over all of its derivations:

$$p_\Lambda(\mathbf{e}|\mathbf{f}) = \sum_{\mathbf{d} \in \Delta(\mathbf{e}, \mathbf{f})} p_\Lambda(\mathbf{d}, \mathbf{e}|\mathbf{f})$$

where $\Delta(\mathbf{e}, \mathbf{f})$ is the set of all derivations of the target sentence $\mathbf{e}$ from the source $\mathbf{f}$.

Here k ranges over the model's features, and $\Lambda = \{\lambda_k\}$ are the model parameters (weights for their corresponding features). The function $q(r, \mathbf{d})$ returns the target ngram context, for a language model with order m, of rule r in derivation $\mathbf{d}$. For a rule which spans the target words (i, j) and $target\_yield(\mathbf{d}) = \{t_0, \cdots, t_l\}$:

$$q(r, \mathbf{d}) = \begin{cases} t_i \cdots t_{i+m-2} \star t_{j-m+2} \cdots t_j & \text{if } j - i > m \\ t_i \cdots t_j & \text{otherwise} \end{cases}$$

The feature functions h_k are real-valued functions over the source and target sentences, and can include overlapping and non-independent features of the data. The features must decompose with the derivation and the ngram context defined by the function q, as shown in Equation (2). The features can reference the entire source sentence coupled with each rule, r, and its target context, in a derivation.

By directly incorporating the language model context q into the model formulation, we will not be able to exactly compute the partition function $Z_\Lambda(\mathbf{f})$, which sums over all possible derivations. Even though a dynamic program over this space would still run in polynomial time, as shown by Chiang (2007), a packed chart representation of the partition function for the binary Hiero grammars used in this work would require $\mathcal{O}(n^3|T|^{4(m-1)})$ space,[1] which is far too large to be practical.

Instead we approximate the partition function using a sum over a large subset of the possible derivations ($\Delta(\mathbf{e},\mathbf{f})$):

$$\begin{aligned} Z_\Lambda(\mathbf{f}) &\approx \sum_{\mathbf{e}} \sum_{\mathbf{d}\in\{\subset\Delta(\mathbf{e},\mathbf{f})\}} \exp \sum_k \lambda_k H_k(\mathbf{d},\mathbf{e},\mathbf{f}) \\ &= \tilde{Z}_\Lambda(\mathbf{f}) \end{aligned}$$

This model formulation raises the questions of what an appropriate *large subset* of derivations for training is, and how to efficiently calculate the sum over all derivations in decoding. In the following sections we elucidate and evaluate our solutions to these problems.

3.1 Sampling Derivations

The training and decoding algorithms presented in the following sections rely upon Monte-Carlo techniques, which in turn require the ability to draw derivation samples from the probability distribution defined by our log-linear model. Here we adapt previously presented algorithms for sampling from a PCFG (Goodman, 1998) for use with our synchronous grammar model. Algorithm 1 describes the algorithm for sampling derivations. The sampling algorithm assumes the pre-existance of a packed chart representation of all derivations for a given source sentence. The *inside algorithm* is then used to calculate the scores needed to define a multinomial distribution over all partial derivations associated with expanding a given child rule. These initial steps are performed once and then an unlimited number of samples can be drawn by calling the recursive SAMPLE procedure. MULTI draws a sample from the distribution over rules for a given chart cell, CHILDREN enumerates the chart cells connected to a rule as variables, and DERIVATION is a recursive tree data structure for derivations. The algorithm is

[1]where $|T|$ is the size of the terminal alphabet, i.e. the number of unique English words.

Algorithm 1 Top-down recursive derivation sampling algorithm.

```
1: procedure SAMPLE($X, i, k$)
2:     $rule$ ← MULTI(inside_chart($X, i, k$))
3:     $c = \phi$
4:     for ($child\_category, x, y$) ∈ CHILDREN($rule$) do
5:         $c \leftarrow c \cup$ SAMPLE($child\_category, x, y$)
6:     end for
7:     return DERIVATION(X, children)
8: end procedure
```

first called on a category and chart cell spanning the entire chart, and then proceeds top down by using the function MULTI to draw the next rule to expand from the distribution defined by the inside scores.

3.2 Approximate Inference

Approximating the partition function with $\tilde{Z}_\Lambda(\mathbf{f})$ could introduce biases into inference and in the following discussion we describe measures taken to minimise the effects of the approximation bias.

An obvious approach to approximating the partition function, and the feature expectations required for calculating the derivative in training, is to use the packed chart of derivations produced by running the cube pruning beam search algorithm of Chiang (2007) on the source sentence. In this case $\tilde{Z}_\Lambda(\mathbf{f})$ includes all the derivations that fall within the cube pruning beam, hopefully representing the majority of the probability mass. We denote the partition function estimated with this cube beam approximation as $\tilde{Z}^{cb}_\Lambda(\mathbf{f})$. This approach has the advantage of using the same beam search dynamic program during training as is used for decoding. As the approximated partition function does not contain all derivations, it is possible that some, or all, of the derivations of the reference translation from the parallel corpus may be excluded. We must therefore intersect the packed chart built from the cube beam with that of the reference derivations to ensure consistency.

Although, as would be done using cube-pruning, it would seem intuitively sensible to approximate the partition function using only high probability derivations, it is possible that doing so will bias our model in odd ways. The space of derivations contained within the beam will be tightly clustered about a maximum, and thus a model trained with such an approximation will only see a very small

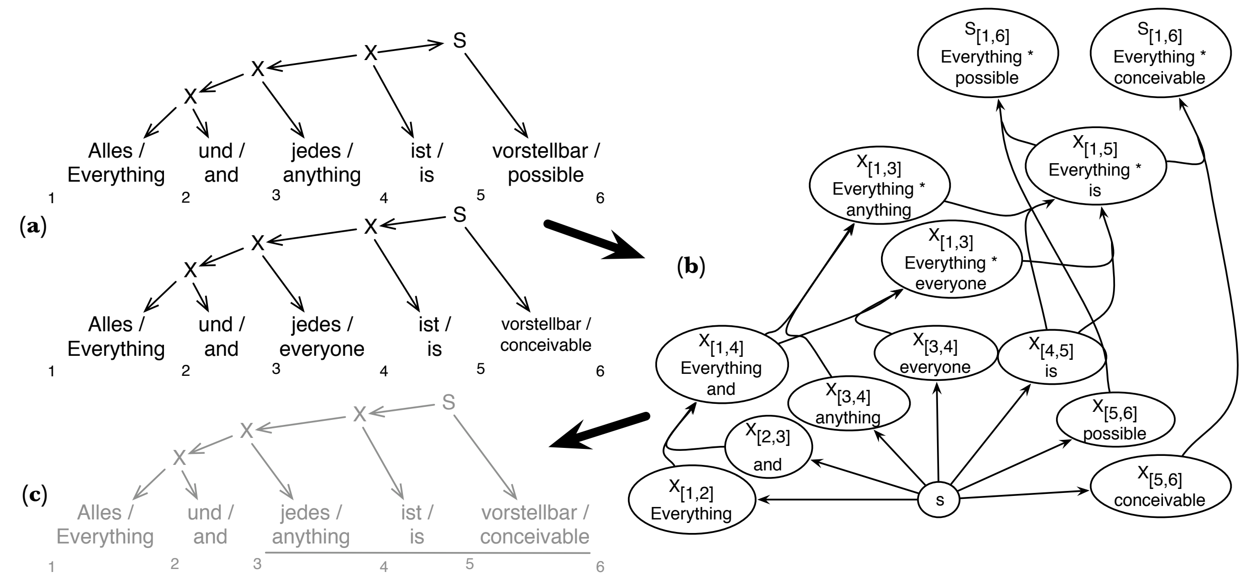


Figure 2. A German-English translation example of building $\tilde{Z}_{\Lambda}^{sam}(\mathbf{f})$ from samples. (a) Two sample derivations are drawn from the model, (b) these samples are then combined into a packed representation, here represented by a hypergraph with target translations elided for a bigram language model. The derivation in (c) is contained within the hypergraph even though it was never explicitly inserted.

part of the overall distribution, possibly leading it astray. Consider the example of a language model feature: as this is a very strong indicator of translation quality, we would expect all derivations within the beam to have a similar (high) language model score, thereby robbing this feature of its discriminating power. However if our model could also see the low probability derivations it would be clear that this feature is indeed very strongly correlated with good translations. Thus a good approximation of the space of derivations is one that includes both good and bad examples, not just a cluster around the maximum.

A principled solution to the problem of approximating the partition function would be to use a Markov Chain Monte Carlo (MCMC) sampler to estimate the sum with a large number of samples. Most of the sampled derivations would be in the high probability region of the distribution, however there would also be a number of samples drawn from the rest of the space, giving the model a more global view of the distribution, avoiding the pitfalls of the narrow view obtained by a beam search. Although effective, the computational cost of such an approach is prohibitive as we would need to draw hundreds of thousands of samples to obtain convergence, for every training iteration.

Here we mediate between the computational advantages of a beam and the broad view of the distribution provided by sampling. Using the algorithm outlined in Section 3.1 we draw samples from the distribution of derivations and then insert these samples into a packed chart representation. This process is illustrated in Figure 2. The packed chart created by intersecting the sample derivations represents a space of derivations much greater than the original samples. In Figure 2 the chart is built from the first two sampled derivations, while the third derivation can be extracted from the chart even though it was never explicitly entered. This approximation of the partition function (denoted $\tilde{Z}_{\Lambda}^{sam}(\mathbf{f})$) allows us to build an efficient packed chart representation of a large number of derivations, centred on those with high probability while still including a significant representation of the low probability space. Derivations corresponding to the reference can be detected during sampling and thus we can build the chart for the reference derivations at the same time as the one approximating the partition function. This could lead to some, or none of, the possible reference derivations being included, as they may not have been sampled. Although we could intersect all of the reference derivations with the sampled chart, this could distort the distribution over derivations, and we believe it to be advantageous to keep the distributions between the partition function and reference charts consistent.

Both of the approximations proposed above, $\tilde{Z}_{\Lambda}^{cb}(\mathbf{f})$ and $\tilde{Z}_{\Lambda}^{sam}(\mathbf{f})$, rely on the pre-existence of a trained translation model in order to either guide the cube-pruning beam, or define the probability distribution from which we draw samples. We solve this chicken and egg problem by first training an exact translation model *without* a language model, and then use this model to create the partition function approximations for training *with* a language model. We denote the distribution without a language model as $p_{\Lambda}^{-LM}(\mathbf{e}|\mathbf{f})$ and that with as $p_{\Lambda}^{+LM}(\mathbf{e}|\mathbf{f})$.

A final training problem that we need to address is the appropriate initialisation of the model parameters. In theory we could simply randomly initialise Λ for $p_{\Lambda}^{+LM}(\mathbf{e}|\mathbf{f})$, however in practice we found that this resulted in poor performance on the development data. This is due to the complex non-convex optimisation function, and the fact that many features will fall outside the approximated charts resulting in random, or zero, weights in testing. We introduce a novel solution in which we use the Gaussian prior over model weights to tie the exact model trained without a language model, which assigns sensible values to all rule features, with the approximated model. The prior over model parameters for $p_{\Lambda}^{+LM}(\mathbf{e}|\mathbf{f})$ is defined as:

$$p^{+LM}(\lambda_k) \propto e^{-\frac{\|\lambda_k - \lambda_k^{-LM}\|^2}{2\sigma^2}}$$

Here we have set the mean parameters of the Gaussian distribution for the approximated model to those learnt for the exact one. This has the effect that any features that fall outside the approximated model will simply retain the weight assigned by the exact model. While for other feature weights the prior will penalise substantial deviations away from Λ^{-LM}, essentially encoding the intuition that the rule rule parameters should not change substantially with the inclusion of language model features.

This results in the following log-likelihood objective and corresponding gradient:

$$\mathcal{L} = \sum_{(\mathbf{e}_i,\mathbf{f}_i)\in\mathcal{D}} \log p_{\Lambda}^{+LM}(\mathbf{e}_i|\mathbf{f}_i) + \sum_k \log p_0^{+LM}(\lambda_k)$$

$$\frac{\partial\mathcal{L}}{\partial\lambda_k} = E_{p_{\Lambda}^{+LM}(\mathbf{d}|\mathbf{e}_i,\mathbf{f}_i)}[h_k] - E_{p_{\Lambda}^{+LM}(\mathbf{e}|\mathbf{f}_i)}[h_k] - \frac{\lambda_k^{+LM} - \lambda_k^{-LM}}{\sigma^2}$$

3.3 Decoding

As stated in Equation 3 the probability of a given translation string is calculated as the sum of the probabilities of all the derivations that yield that string. In decoding, where the reference translation is not known, the exact calculation of this summation is NP-Hard. This problem also arises in monolingual parsing with probabilistic tree substitution grammars and has been tackled in the literature using Monte-Carlo sampling methods (Chappelier and Rajman, 2000). Their approach is directly applicable to our SCFG decoding problem and we can use Algorithm 1 to draw sample translation derivations for the source sentence. The probability of a translation can be calculated simply from the number of times a derivation that yields it was sampled, divided by the total number of samples. For the $p_{\Lambda}^{-LM}(\mathbf{e}|\mathbf{f})$ model we can build the full chart of all possible derivations and thus sample from the true distribution over derivations. For the $p_{\Lambda}^{+LM}(\mathbf{e}|\mathbf{f})$ model we suffer the same problem as in training and cannot build the full chart. Instead a chart is built using the cube-pruning algorithm with a wide beam and we then draw samples from this chart. Although sampling from a reduced chart will result in biased samples, in Section 4 we show this approach to be effective in practice.[2] In Section 4 we compare our sampling approach to the heuristic beam search proposed by Blunsom et al. (2008).

It is of interest to compare our proposed decoding algorithms to minimum Bayes risk (MBR) decoding (Kumar and Byrne, 2004), a commonly used decoding method. From a theoretical standpoint, the summing of derivations for a given translation is exactly

[2]We have experimented with using a Metropolis Hastings sampler, with $p_{\Lambda}^{-LM}(\mathbf{e}|\mathbf{f})$ as the proposal distribution, to sample from the true distribution with the language model. Unfortunately the sample rejection rate was very high such that this method proved infeasibly slow.

equivalent to performing MBR with a 0/1 loss function over derivations. From a practical perspective, MBR is normally performed with $\textsc{Bleu}$ as the loss and approximated using n-best lists. These n-best lists are produced using algorithms tuned to remove multiple derivations of the same translation (which have previously been seen as undesirable). However, it would be simple to extend our sampling based decoding algorithm to calculate the MBR estimate using $\textsc{Bleu}$, in theory providing a lower variance estimate than attained with n-best lists.

4 Evaluation

We evaluate our model on the IWSLT 2005 Chinese to English translation task (Eck and Hori, 2005), using the 2004 test set as development data for tuning the hyperparameters and MERT training the benchmark systems. The statistics for this data are presented in Table 1.[3] The training data made available for this task consisted of 40k pairs of transcribed utterances, drawn from the travel domain. The development and test data for this task are somewhat unusual in that each sentence has a single human translated reference, and fifteen paraphrases of this reference, provided by monolingual annotators. Model performance is evaluated using the standard $\textsc{Bleu}$ metric (Papineni et al., 2002) which measures average n-gram precision, $n \leq 4$, and we use the NIST definition of the brevity penalty for multiple reference test sets. We provide evaluation against both the entire multi-reference sets, and the single human translation.

Our translation grammar is induced using the standard alignment and rule extraction heuristics used in hierarchical translation models (Chiang, 2007).[4] As these heuristics aren't based on a generative model, and don't guarantee that the target translation will be reachable from the source, we discard those sentence pairs for which we cannot produce a derivation, leaving 38,405 sentences for training.

Our base model contains a single feature for each rule which counts the number of times it appeared in a particular derivation. For models which include a language model, we train a standard Kneser-Ney trigram model on the target side of the training corpus. We also include a word penalty feature to compensate for the shortening effect of the language model. In total our model contains 2.9M features.

The aims of our evaluation are: (1) to determine that our proposed training regimes are able to realise performance increase when training sparse rule features and a real valued language model feature together, (2) that the model is able to effectively use rich features over the source sentence, and (3) to confirm that our model obtains performance competitive with the current state-of-the-art.

4.1 Inference and Decoding

We have described a number of modelling choices which aim to compensate for the training biases introduced by incorporating a language model feature through approximate inference. Our a priori knowledge from other SMT systems suggests that incorporating a language model should lead to large increases in $\textsc{Bleu}$ score. In this evaluation we aim to determine whether our training regimes are able to realises these expected gains.

Table 2 shows a matrix of development $\textsc{Bleu}$ scores achieved by varying the approximation of the partition function in training, and varying the decoding algorithm. If we consider the vertical axis we can see that the sampling method for approximating the partition function has a small but consistent advantage over using the cube-pruning beam. The charts produced by the sampling approach occupy roughly half the disc space as those produced by the beam search, so in subsequent experiments we present results using the $\tilde{Z}^{sam}_{\Lambda}(\mathbf{f})$ approximation.

Comparing the decoding algorithms on the horizontal axis we can reconfirm the findings of Blunsom et al. (2008) that the max-translation decoding outperforms the Viterbi max-derivation approximation. It is also of note that this $\textsc{Bleu}$ increase is robust to the introduction of the language model feature, assuaging fears that the max-translation approach may have been doing the job of the language model. We also compare using Monte-Carlo sampling for decoding with the previously proposed heuristic beam search algorithm. The difference between the two algorithms is small, however

[3]Development and test set statistics are for the single human translation reference.

[4]With the exception that we allow unaligned words at the boundary of rules. This improves training set coverage.

	Training		Development		Test	
	Chinese	English	Chinese	English	Chinese	English
Utterances	38405		500		506	
Segments/Words	317621	353116	3464	3752	3784	3823
Av. Utterances Length	8	9	6	7	7	7
Longest Utterance	55	68	58	62	61	56

Table 1. IWSLT Chinese to English translation corpus statistics.

Model	Max-derivation	Max-translation(Beam)	Max-translation(Sampling)
$p_\Lambda^{-LM}(\mathbf{e}\|\mathbf{f})$	31.0	32.5	32.6
$p_\Lambda^{+LM}(\mathbf{e}\|\mathbf{f})$ $(\tilde{Z}_\Lambda^{cb}(\mathbf{f}))$	39.1	39.8	39.8
$p_\Lambda^{+LM}(\mathbf{e}\|\mathbf{f})$ $(\tilde{Z}_\Lambda^{sam}(\mathbf{f}))$	39.9	40.5	40.6

Table 2. Development set results for varying the approximation of the partition function in training, $\tilde{Z}_\Lambda^{cb}(\mathbf{f})$ vs. $\tilde{Z}_\Lambda^{sam}(\mathbf{f})$, and decoding using the Viterbi max-derivation algorithm, or the max-translation algorithm with either a beam approximation or Monte-Carlo sampling.

we feeling the sampling approach is more theoretically justified and adopt it for our later experiments.

The most important result from this evaluation is that both our training regimes realise substantial gains from the introduction of the language model feature. Thus we can be confident that our model is capable of modelling the distribution over translations even when the space over all derivations is intractable to dynamically program exactly.

4.2 A Discriminative Syntactic Translation Model

In the previous sections we've described and evaluated a statistical model of translation that is able to estimate a probability distribution over translations using millions of sparse features. A prime motivation for such a model is the ability to define complex features over more than just the surface forms of the source and target strings. There are limitless options for such features, and previous work has focused on defining token based features such as part-of-speech and morphology (Ittycheriah and Roukos, 2007). Although such features are applicable to our model, here we attempt to test the model's ability to incorporate complex features over source-side syntax trees, essentially subsuming and extending previous work on tree-to-string translation models (Huang et al., 2006; Mi et al., 2008).

We first parse the source side of our training, development and test corpora using the Stanford parser.[5] Next, while building the synchronous charts required for training, whenever a rule is used in a derivation a feature is activated which captures: (1) the constituent spanning the rule's source side in the syntax tree (if any) (2) constituents spanning any variables in the rule, and (3) the rule's target side surface form. Figure 3 illustrates this process.

These syntactic features are equivalent to the grammar rules extracted for tree-to-string translation systems. The key difference in our model is that the source syntax tree is treated as conditioning context and it's information encoded as features, thus this information can be used or ignored as the model sees fit. This avoids the problems associated with explicitly encoding the source syntax in the grammar, such as sparsity and overly constraining the model. In addition we could easily incorporate features over multiple source trees, for example mixing labelled syntax trees with dependency graphs.

We limit the extraction of syntactic features to those that appear in at least two training derivations, giving a total of 4.2M syntactic features, for an overall total of 7.1M features.

4.3 Discussion

Table 3 shows the results from applying our described models to the test set. We benchmark our results against a model (Hiero) which was directly trained to optimise $\textsc{Bleu}^{NIST}$ using the standard MERT algorithm (Och, 2003) and the full set of translation and lexical weight features described for the Hiero model (Chiang, 2007). As well as

[5]http://nlp.stanford.edu/software/lex-parser.shtml

Model	$\textsc{Bleu}^{NIST}$	$\textsc{Bleu}^{IBM}$	$\textsc{Bleu}^{HumanRef}$
$p_\Lambda^{-LM}(\mathbf{e}\|\mathbf{f})$	33.5	35.2	25.2
$p_\Lambda^{+LM}(\mathbf{e}\|\mathbf{f})$	44.6	44.6	31.2
$p_\Lambda^{+LM}(\mathbf{e}\|\mathbf{f})$ + syntax	45.3	45.2	31.8
MERT ($\textsc{Bleu}^{NIST}$)	46.2	44.5	30.2

Table 3. Test set results.

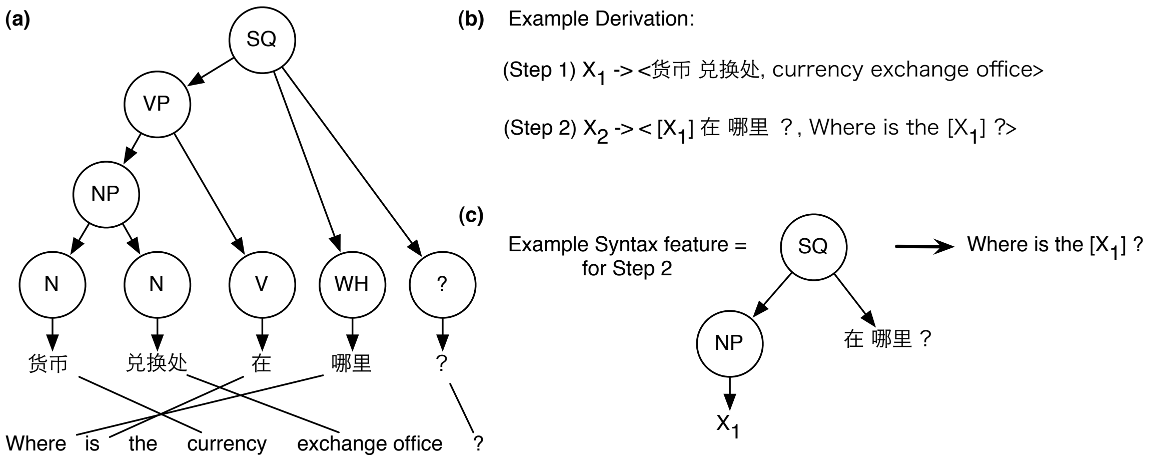

(a)
SQ
VP
NP
N
N
V
WH
?
货币
兑换处
在
哪里
?
Where
is
the
currency
exchange office
?
(b) Example Derivation:
(Step 1) X1 -> <货币 兑换处, currency exchange office>
(Step 2) X2 -> < [X1] 在 哪里 ?, Where is the [X1] ?>
(c)
Example Syntax feature = for Step 2
SQ
Where is the [X1] ?
NP
在 哪里 ?
X1

Figure 3. Syntax feature example: For the parsed source and candidate translation (a), with the derivation (b), we extract the syntax feature in (c) by combining the grammar rule with the source syntax of the constituents contained within that rule.

Source	我 身 上 没有 足够 的 钱 去 买 一 张 新 的 飞机 票 。
$p_\Lambda^{-LM}(\mathbf{e}\|\mathbf{f})$	don 't have enough bag on me change please go purchase a new by plane .
$p_\Lambda^{+LM}(\mathbf{e}\|\mathbf{f})$	i have enough money to buy a new one by air .
$p_\Lambda^{+LM}(\mathbf{e}\|\mathbf{f})$ + syntax	i don 't have enough money to buy a new airline ticket .
MERT ($\textsc{Bleu}^{NIST}$)	i don 't have enough money to buy a new ticket .
Reference	i do n't have enough money with me to buy a new airplane ticket .

Table 4. Example test set output produced when: not using a language model, using a language model, also using syntax, output optimised using MERT and finally the reference

$\textsc{Bleu}^{NIST}$ (brevity penalty uses the shortest reference), we also include results from the IBM ($\textsc{Bleu}^{IBM}$) metric (brevity penalty uses the closest reference), and using only the actual human translation in the test set, not the monolingual paraphrase multiple references ($\textsc{Bleu}^{HumanRef}$).

The first result of interest is that we see an increase in performance through the incorporation of the source syntax features. This is an encouraging result as the transcribed speech source sentences are well out of the domain of the data on which the parser was trained, suggesting that our model is able to sift the good information from the noisy in the unreliable source syntax trees. Table 4 shows illustrative system output on the test set.

On the $\textsc{Bleu}^{NIST}$ metric we see that our models under-perform the MERT trained system. We hypothesise that this is predominately due to the interaction of the brevity penalty with the unusual nature of the multiple paraphrase reference training and development data. Their performance is however quite consistent across the different interpretations of the brevity penalty (NIST vs. IBM). This contrasts with the MERT trained model, which clearly over-fits to the NIST metric that it was trained on and underperforms our models when evaluated on the single human test translations. If we directly compare the brevity penalties of the MERT model (0.868) and our discriminative model incorporating source syntax (0.942), on the these single references, we can see that the MERT training has optimised to the shortest paraphrase reference.

From these results it is difficult to draw any hard conclusions on the relative performance of the different training regimes. However we feel confident in claiming that we have achieved our goal of training a probabilistic model on millions of sparse features which obtains performance competitive with the current state-of-the-art training algorithm.

5 Conclusion

In this paper we have shown that statistical machine translation can be effectively modelled as a well posed machine learning task. In doing so we have described a model capable of estimating a probability distribution over translations using sparse complex features, and achieving performance comparable to the state-of-the-art on standard metrics. With further work on scaling these models to large data sets, and engineering high performance features, we believe this research has the potential to provide significant increases in translation quality.

Acknowledgements

The authors acknowledge the support of the EPSRC grant EP/D074959/1.

References

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proc. of the 46th Annual Conference of the Association for Computational Linguistics: Human Language Technologies (ACL-08:HLT)*, pages 200–208, Columbus, Ohio, June.

Jean-Cédric Chappelier and Martin Rajman. 2000. Monte-carlo sampling for np-hard maximization problems in the framework of weighted parsing. In *NLP '00: Proceedings of the Second International Conference on Natural Language Processing*, pages 106–117, London, UK.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Matthias Eck and Chiori Hori. 2005. Overview of the IWSLT 2005 evaluation campaign. In *Proc. of the International Workshop on Spoken Language Translation*, Pittsburgh, October.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proc. of the 44th Annual Meeting of the ACL and 21st International Conference on Computational Linguistics (COLING/ACL-2006)*, pages 961–968, Sydney, Australia, July.

Joshua T. Goodman. 1998. *Parsing inside-out*. Ph.D. thesis, Cambridge, MA, USA. Adviser-Stuart Shieber.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *In Proceedings of the 7th Biennial Conference of the Association for Machine Translation in the Americas (AMTA)*, Boston, MA.

Abraham Ittycheriah and Salim Roukos. 2007. Direct translation model 2. In *Proc. of the 7th International Conference on Human Language Technology Research and 8th Annual Meeting of the NAACL (HLT-NAACL 2007)*, pages 57–64, Rochester, USA.

Shankar Kumar and William Byrne. 2004. Minimum bayes-risk decoding for statistical machine translation. In *Proc. of the 4th International Conference on Human Language Technology Research and 5th Annual Meeting of the NAACL (HLT-NAACL 2004)*, pages 169–176.

Philip M. Lewis II and Richard E. Stearns. 1968. Syntax-directed transduction. *J. ACM*, 15(3):465–488.

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

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proc. of the 46th Annual Conference of the Association for Computational Linguistics: Human Language Technologies (ACL-08:HLT)*, pages 192–199, Columbus, Ohio, June.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of the 41st Annual Meeting of the ACL (ACL-2003)*, pages 160–167, Sapporo, Japan.

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 Meeting of the ACL and 3rd Annual Meeting of the NAACL (ACL-2002)*, pages 311–318, Philadelphia, Pennsylvania.

Taro Watanabe, Hajime Tsukada, and Hideki Isozaki. 2006. Left-to-right target generation for hierarchical phrase-based translation. In *Proc. of the 44th Annual Meeting of the ACL and 21st International Conference on Computational Linguistics (COLING/ACL-2006)*, pages 777–784, Sydney, Australia.

Online Large-Margin Training of Syntactic and Structural Translation Features

David Chiang
Information Sciences Institute
University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292, USA
chiang@isi.edu

Yuval Marton and **Philip Resnik**
Department of Linguistics and the UMIACS
Laboratory for Computational Linguistics
and Information Processing
University of Maryland
College Park, MD 20742, USA
{ymarton,resnik}@umiacs.umd.edu

Abstract

Minimum-error-rate training (MERT) is a bottleneck for current development in statistical machine translation because it is limited in the number of weights it can reliably optimize. Building on the work of Watanabe et al., we explore the use of the MIRA algorithm of Crammer et al. as an alternative to MERT. We first show that by parallel processing and exploiting more of the parse forest, we can obtain results using MIRA that match or surpass MERT in terms of both translation quality and computational cost. We then test the method on two classes of features that address deficiencies in the Hiero hierarchical phrase-based model: first, we simultaneously train a large number of Marton and Resnik's soft syntactic constraints, and, second, we introduce a novel structural distortion model. In both cases we obtain significant improvements in translation performance. Optimizing them in combination, for a total of 56 feature weights, we improve performance by 2.6 Bleu on a subset of the NIST 2006 Arabic-English evaluation data.

1 Introduction

Since its introduction by Och (2003), minimum error rate training (MERT) has been widely adopted for training statistical machine translation (MT) systems. However, MERT is limited in the number of feature weights that it can optimize reliably, with folk estimates of the limit ranging from 15 to 30 features.

One recent example of this limitation is a series of experiments by Marton and Resnik (2008), in which they added syntactic features to Hiero (Chiang, 2005; Chiang, 2007), which ordinarily uses no linguistically motivated syntactic information. Each of their new features rewards or punishes a derivation depending on how similar or dissimilar it is to a syntactic parse of the input sentence. They found that in order to obtain the greatest improvement, these features had to be specialized for particular syntactic categories and weighted independently. Not being able to optimize them all at once using MERT, they resorted to running MERT many times in order to test different combinations of features. But it would have been preferable to use a training method that can optimize the features all at once.

There has been much work on improving MERT's performance (Duh and Kirchoff, 2008; Smith and Eisner, 2006; Cer et al., 2008), or on replacing MERT wholesale (Turian et al., 2007; Blunsom et al., 2008). This paper continues a line of research on online discriminative training (Tillmann and Zhang, 2006; Liang et al., 2006; Arun and Koehn, 2007), extending that of Watanabe et al. (2007), who use the Margin Infused Relaxed Algorithm (MIRA) due to Crammer et al. (2003; 2006). Our guiding principle is practicality: like Watanabe et al., we train on a small tuning set comparable in size to that used by MERT, but by parallel processing and exploiting more of the parse forest, we obtain results using MIRA that match or surpass MERT in terms of both translation quality and computational cost on a large-scale translation task.

Taking this further, we test MIRA on two classes of features that make use of syntactic information and hierarchical structure. First, we generalize Marton and Resnik's (2008) soft syntactic constraints by training all of them simultaneously; and, second, we introduce a novel structural distortion model. We obtain significant improvements in both cases, and further large improvements when the two feature sets are combined.

The paper proceeds as follows. We describe our training algorithm in section 2; our generalization of Marton and Resnik's soft syntactic constraints in section 3; our novel structural distortion features in section 4; and experimental results in section 5.

2 Learning algorithm

The translation model is a standard linear model (Och and Ney, 2002), which we train using MIRA (Crammer and Singer, 2003; Crammer et al., 2006), following Watanabe et al. (2007). We describe the basic algorithm first and then progressively refine it.

2.1 Basic algorithm

Let $\mathbf{e}$, by abuse of notation, stand for both output strings and their derivations. We represent the feature vector for derivation $\mathbf{e}$ as $\mathbf{h}(\mathbf{e})$. Initialize the feature weights $\mathbf{w}$. Then, repeatedly:

- Select a batch of input sentences $\mathbf{f}_1, \ldots, \mathbf{f}_m$.
- Decode each $\mathbf{f}_i$ to obtain a set of hypothesis translations $\mathbf{e}_{i1}, \ldots, \mathbf{e}_{in}$.
- For each i, select one of the $\mathbf{e}_{ij}$ to be the *oracle translation* $\mathbf{e}_i^*$, by a criterion described below. Let $\Delta\mathbf{h}_{ij} = \mathbf{h}(\mathbf{e}_i^*) - \mathbf{h}(\mathbf{e}_{ij})$.
- For each $\mathbf{e}_{ij}$, compute the *loss* ℓ_{ij}, which is some measure of how bad it would be to guess $\mathbf{e}_{ij}$ instead of $\mathbf{e}_i^*$.
- Update $\mathbf{w}$ to the value of $\mathbf{w}'$ that minimizes:

$$\frac{1}{2}\|\mathbf{w}' - \mathbf{w}\|^2 + C\sum_{i=1}^{m} \max_{1\leq j\leq n}(\ell_{ij} - \Delta\mathbf{h}_{ij} \cdot \mathbf{w}') \quad (1)$$

  where we set $C = 0.01$. The first term means that we want $\mathbf{w}'$ to be close to $\mathbf{w}$, and second term (the *generalized hinge loss*) means that we want $\mathbf{w}'$ to score $\mathbf{e}_i^*$ higher than each $\mathbf{e}_{ij}$ by a margin at least as wide as the loss ℓ_{ij}.

When training is finished, the weight vectors from all iterations are averaged together. (If multiple passes through the training data are made, we only average the weight vectors from the last pass.) The technique of averaging was introduced in the context of perceptrons as an approximation to taking a vote among all the models traversed during training, and has been shown to work well in practice (Freund and Schapire, 1999; Collins, 2002). We follow McDonald et al. (2005) in applying this technique to MIRA.

Note that the objective (1) is not the same as that used by Watanabe et al.; rather, it is the same as that used by Crammer and Singer (2003) and related to that of Taskar et al. (2005). We solve this optimization problem using a variant of sequential minimal optimization (Platt, 1998): for each i, initialize $\alpha_{ij} = C$ for a single value of j such that $\mathbf{e}_{ij} = \mathbf{e}_i^*$, and initialize $\alpha_{ij} = 0$ for all other values of j. Then, repeatedly choose a sentence i and a pair of hypotheses j, j', and let

$$\mathbf{w}' \leftarrow \mathbf{w}' + \delta(\Delta\mathbf{h}_{ij} - \Delta\mathbf{h}_{ij'}) \quad (2)$$

$$\alpha_{ij} \leftarrow \alpha_{ij} + \delta \quad (3)$$

$$\alpha_{ij'} \leftarrow \alpha_{ij'} - \delta \quad (4)$$

where

$$\delta = \underset{[-\alpha_{ij},\alpha_{ij'}]}{\text{clip}} \frac{(\ell_{ij} - \ell_{ij'}) - (\Delta\mathbf{h}_{ij} - \Delta\mathbf{h}_{ij'}) \cdot \mathbf{w}'}{\|\Delta\mathbf{h}_{ij} - \Delta\mathbf{h}_{ij'}\|^2} \quad (5)$$

where the function $\text{clip}_{[x,y]}(z)$ gives the closest number to z in the interval $[x, y]$.

2.2 Loss function

Assuming Bleu as the evaluation criterion, the loss ℓ_{ij} of $\mathbf{e}_{ij}$ relative to $\mathbf{e}_i^*$ should be related somehow to the difference between their Bleu scores. However, Bleu was not designed to be used on individual sentences; in general, the highest-Bleu translation of a sentence depends on what the other sentences in the test set are. Sentence-level approximations to Bleu exist (Lin and Och, 2004; Liang et al., 2006), but we found it most effective to perform Bleu computations in the context of a set O of previously-translated sentences, following Watanabe et al. (2007). However, we don't try to accumulate translations for the entire dataset, but simply maintain an exponentially-weighted moving average of previous translations.

More precisely: For an input sentence $\mathbf{f}$, let $\mathbf{e}$ be some hypothesis translation and let $\{\mathbf{r}_k\}$ be the set of reference translations for $\mathbf{f}$. Let $\mathbf{c}(\mathbf{e}; \{\mathbf{r}_k\})$, or simply $\mathbf{c}(\mathbf{e})$ for short, be the vector of the following counts: $|\mathbf{e}|$, the effective reference length $\min_k |\mathbf{r}_k|$, and, for $1 \leq n \leq 4$, the number of n-grams in $\mathbf{e}$, and the number of n-gram matches between $\mathbf{e}$ and $\{\mathbf{r}_k\}$. These counts are sufficient to calculate a Bleu score, which we write as $\text{Bleu}(\mathbf{c}(\mathbf{e}))$. The pseudo-document O is an exponentially-weighted moving average of these vectors. That is, for each training sentence, let $\hat{\mathbf{e}}$ be the 1-best translation; after processing the sentence, we update O, and its input length O_f:

$$O \leftarrow 0.9(O + \mathbf{c}(\hat{\mathbf{e}})) \tag{6}$$

$$O_f \leftarrow 0.9(O_f + |\mathbf{f}|) \tag{7}$$

We can then calculate the Bleu score of hypotheses $\mathbf{e}$ in the context of O. But the larger O is, the smaller the impact the current sentence will have on the Bleu score. To correct for this, and to bring the loss function roughly into the same range as typical margins, we scale the Bleu score by the size of the input:

$$B(\mathbf{e}; \mathbf{f}, \{\mathbf{r}_k\}) = (O_f + |\mathbf{f}|) \times \text{Bleu}(O + \mathbf{c}(\mathbf{e}; \{\mathbf{r}_k\})) \tag{8}$$

which we also simply write as $B(\mathbf{e})$. Finally, the loss function is defined to be:

$$\ell_{ij} = B(\mathbf{e}_i^*) - B(\mathbf{e}_{ij}) \tag{9}$$

2.3 Oracle translations

We now describe the selection of $\mathbf{e}^*$. We know of three approaches in previous work. The first is to force the decoder to output the reference sentence exactly, and select the derivation with the highest model score, which Liang et al. (2006) call *bold updating*. The second uses the decoder to search for the highest-Bleu translation (Tillmann and Zhang, 2006), which Arun and Koehn (2007) call *max*-Bleu *updating*. Liang et al. and Arun and Koehn experiment with these methods and both opt for a third method, which Liang et al. call *local updating*: generate an n-best list of translations and select the highest-Bleu translation from it. The intuition is that due to noise in the training data or reference translations, a high-Bleu translation may actually use peculiar rules which it would be undesirable to encourage the model to use. Hence, in local updating, the search for the highest-Bleu translation is limited to the n translations with the highest model score, where n must be determined experimentally.

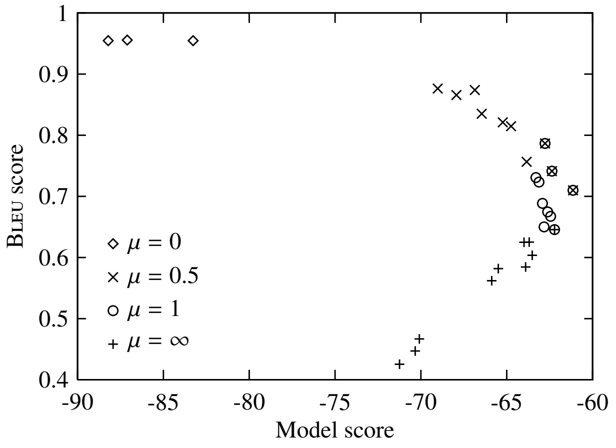


Figure 1: Scatter plot of 10-best unique translations of a single sentence obtained by forest rescoring using various values of μ in equation (11).

Here, we introduce a new oracle-translation selection method, formulating the intuition behind local updating as an optimization problem:

$$\mathbf{e}^* = \arg\max_{\mathbf{e}} (B(\mathbf{e}) + \mathbf{h}(\mathbf{e}) \cdot \mathbf{w}) \tag{10}$$

Instead of choosing the highest-Bleu translation from an n-best list, we choose the translation that maximizes a combination of (approximate) Bleu and the model.

We can also interpret (10) in the following way: we want $\mathbf{e}^*$ to be the max-Bleu translation, but we also want to minimize (1). So we balance these two criteria against each other:

$$\mathbf{e}^* = \arg\max_{\mathbf{e}} (B(\mathbf{e}) - \mu(B(\mathbf{e}) - \mathbf{h}(\mathbf{e}) \cdot \mathbf{w})) \tag{11}$$

where $(B(\mathbf{e}) - \mathbf{h}(\mathbf{e}) \cdot \mathbf{w})$ is that part of (1) that depends on $\mathbf{e}^*$, and μ is a parameter that controls how much we are willing to allow some translations to have higher Bleu than $\mathbf{e}^*$ if we can better minimize (1). Setting $\mu = 0$ would reduce to max-Bleu updating; setting $\mu = \infty$ would never update $\mathbf{w}$ at all. Setting $\mu = 0.5$ reduces to equation (10).

Figure 1 shows the 10-best unique translations for a single input sentence according to equation (11) under various settings of μ. The points at far right are the translations that are scored highest according to the model. The $\mu = 0$ points in the upper-left corner are typical of oracle translations that would be selected under the max-BLEU policy: they indeed have a very high BLEU score, but are far removed from the translations preferred by the model; thus they would cause violent updates to $\mathbf{w}$. Local updating would select the topmost point labeled $\mu = 1$. Our scheme would select one of the $\mu = 0.5$ points, which have BLEU scores almost as high as the max-BLEU translations, yet are not very far from the translations preferred by the model.

2.4 Selecting hypothesis translations

What is the set $\{\mathbf{e}_{ij}\}$ of translation hypotheses? Ideally we would let it be the set of all possible translations, and let the objective function (1) take all of them into account. This is the approach taken by Taskar et al. (2004), but their approach assumes that the loss function can be decomposed into local loss functions. Since our loss function cannot be so decomposed, we select:

- the 10-best translations according to the model;

we then rescore the forest to obtain

- the 10-best translations according to equation (11) with $\mu = 0.5$, the first of which is the oracle translation, and
- the 10-best translations with $\mu = \infty$, to serve as negative examples.

The last case is what Crammer et al. (2006) call *max-loss updating* (where "loss" refers to the generalized hinge loss) and Taskar et al. (2005) call *loss-augmented inference*. The rationale here is that since the objective (1) tries to minimize $\max_j(\ell_{ij} - \Delta\mathbf{h}_{ij} \cdot \mathbf{w}')$, we should include the translations that have the highest $(\ell_{ij} - \Delta\mathbf{h}_{ij} \cdot \mathbf{w})$ in order to approximate the effect of using the whole forest.

See Figure 1 again for an illustration of the hypotheses selected for a single sentence. The max-BLEU points in the upper left are not included (and would have no effect even if they were included). The $\mu = \infty$ points in the lower-right are the negative examples: they are poor translations that are scored too high by the model, and the learning algorithm attempts to shift them to the left.

To perform the forest rescoring, we need to use several approximations, since an exact search for BLEU-optimal translations is NP-hard (Leusch et al., 2008). For every derivation $\mathbf{e}$ in the forest, we calculate a vector $\mathbf{c}(\mathbf{e})$ of counts as in Section 2.2 except using *unclipped* counts of n-gram matches (Dreyer et al., 2007), that is, the number of matches for an n-gram can be greater than the number of occurrences of the n-gram in any reference translation. This can be done efficiently by calculating $\mathbf{c}$ for every hyperedge (rule application) in the forest:

- the number of output words generated by the rule
- the effective reference length scaled by the fraction of the input sentence consumed by the rule
- the number of n-grams formed by the application of the rule ($1 \leq n \leq 4$)
- the (unclipped) number of n-gram matches formed by the application of the rule ($1 \leq n \leq 4$)

We keep track of n-grams using the same scheme used to incorporate an n-gram language model into the decoder (Wu, 1996; Chiang, 2007).

To find the best derivation in the forest, we traverse it bottom-up as usual, and for every set of alternative subtranslations, we select the one with the highest score. But here a rough approximation lurks, because we need to calculate B on the nodes of the forest, but B does not have the optimal substructure property, i.e., the optimal score of a parent node cannot necessarily be calculated from the optimal scores of its children. Nevertheless, we find that this rescoring method is good enough for generating high-BLEU oracle translations and low-BLEU negative examples.

2.5 Parallelization

One convenient property of MERT is that it is embarrassingly parallel: we decode the entire tuning set sending different sentences to different processors, and during optimization of feature weights, different random restarts can be sent to different processors. In order to make MIRA comparable in efficiency to MERT, we must parallelize it. But with an online learning algorithm, parallelization requires a little more coordination. We run MIRA on each processor simultaneously, with each maintaining its own weight vector. A master process distributes different sentences from the tuning set to each of the processors; when each processor finishes decoding a sentence, it transmits the resulting hypotheses, with their losses, to all the other processors and receives any hypotheses waiting from other processors. Those hypotheses were generated from different weight vectors, but can still provide useful information. The sets of hypotheses thus collected are then processed as one batch. When the whole training process is finished, we simply average all the weight vectors from all the processors.

Having described our training algorithm, which includes several practical improvements to Watanabe et al.'s usage of MIRA, we proceed in the remainder of the paper to demonstrate the utility of the our training algorithm on models with large numbers of structurally sensitive features.

3 Soft syntactic constraints

The first features we explore are based on a line of research introduced by Chiang (2005) and improved on by Marton and Resnik (2008). A hierarchical phrase-based translation model is based on synchronous context-free grammar, but does not normally use any syntactic information derived from linguistic knowledge or treebank data: it uses translation rules that span any string of words in the input sentence, without regard for parser-defined syntactic constituency boundaries. Chiang (2005) experimented with a constituency feature that rewarded rules whose source language side exactly spans a syntactic constituent according to the output of an external source-language parser. This feature can be viewed as a *soft syntactic constraint*: it biases the model toward translations that respect syntactic structure, but does not force it to use them. However, this more syntactically aware model, when tested in Chinese-English translation, did not improve translation performance.

Recently, Marton and Resnik (2008) revisited the idea of constituency features, and succeeded in showing that finer-grained soft syntactic constraints yield substantial improvements in BLEU score for both Chinese-English and Arabic-English translation. In addition to adding separate features for different syntactic nonterminals, they introduced a new type of constraint that penalizes rules when the source language side *crosses* the boundaries of a source syntactic constituent, as opposed to simply rewarding rules when they are consistent with the source-language parse tree.

Marton and Resnik optimized their features' weights using MERT. But since MERT does not scale well to large numbers of feature weights, they were forced to test individual features and manually selected feature combinations each in a separate model. Although they showed gains in translation performance for several such models, many larger, potentially better feature combinations remained unexplored. Moreover, the best-performing feature subset was different for the two language pairs, suggesting that this labor-intensive feature selection process would have to be repeated for each new language pair.

Here, we use MIRA to optimize Marton and Resnik's finer-grained single-category features all at once. We define below two sets of features, a coarse-grained class that combines several constituency categories, and a fine-grained class that puts different categories into different features. Both kinds of features were used by Marton and Resnik, but only a few at a time. Crucially, our training algorithm provides the ability to train all the fine-grained features, a total of 34 feature weights, simultaneously.

Coarse-grained features As the basis for coarse-grained syntactic features, we selected the following nonterminal labels based on their frequency in the tuning data, whether they frequently cover a span of more than one word, and whether they represent linguistically relevant constituents: NP, PP, S, VP, SBAR, ADJP, ADVP, and QP. We define two new features, one which fires when a rule's source side span in the input sentence matches any of the above-mentioned labels in the input parse, and another which fires when a rule's source side span crosses a boundary of one of these labels (e.g., its source side span only partially covers the words in a VP subtree, and it also covers some or all or the words outside the VP subtree). These two features are equivalent to Marton and Resnik's $\mathrm{XP}^{=}$ and XP^{+} feature combinations, respectively.

Fine-grained features We selected the following nonterminal labels that appear more than 100 times in the tuning data: NP, PP, S, VP, SBAR, ADJP, WHNP, PRT, ADVP, PRN, and QP. The labels that were excluded were parts of speech, nonconstituent labels like FRAG, or labels that occurred only two or three times. For each of these labels X, we added a separate feature that fires when a rule's source side span in the input sentence matches X, and a second feature that fires when a span crosses a boundary of X. These features are similar to Marton and Resnik's $X^{=}$ and X^{+}, except that our set includes features for WHNP, PRT, and PRN.

4 Structural distortion features

In addition to parser-based syntactic constraints, which were introduced in prior work, we introduce a completely new set of features aimed at improving the modeling of reordering within Hiero. Again, the feature definition gives rise to a larger number of features than one would expect to train successfully using MERT.

In a phrase-based model, reordering is performed both within phrase pairs and by the phrase-reordering model. Both mechanisms are able to learn that longer-distance reorderings are more costly than shorter-distance reorderings: phrase pairs, because phrases that involve more extreme reorderings will (presumably) have a lower count in the data, and phrase reordering, because models are usually explicitly dependent on distance.

By contrast, in a hierarchical model, all reordering is performed by a single mechanism, the rules of the grammar. In some cases, the model will be able to learn a preference for shorter-distance reorderings, as in a phrase-based system, but in the case of a word being reordered across a nonterminal, or two nonterminals being reordered, there is no dependence in the model on the size of the nonterminal or nonterminals involved in reordering.

So, for example, if we have rules

$$X \rightarrow \langle \text{il dit } X_1, \text{he said } X_1 \rangle \tag{12}$$

$$X \rightarrow \langle \text{il dit } X_1, X_1 \text{ he said} \rangle \tag{13}$$

we might expect that rule (12) is more common in general, but that rule (13) becomes more and more rare as X_1 gets larger. The default Hiero features have no way to learn this.

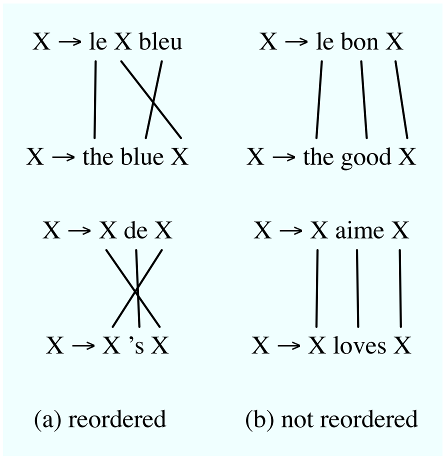


Figure 2: Classifying nonterminal occurrences for the structural distortion model.

To address this defect, we can classify every nonterminal pair occurring on the right-hand side of each grammar rule as "reordered" or "not reordered", that is, whether it intersects any other word alignment link or nonterminal pair (see Figure 2). We then define coarse- and fine-grained versions of the structural distortion model.

Coarse-grained features Let R be a binary-valued random variable that indicates whether a nonterminal occurrence is reordered, and let S be an integer-valued random variable that indicates how many source words are spanned by the nonterminal occurrence. We can estimate $P(R \mid S)$ via relative-frequency estimation from the rules as they are extracted from the parallel text, and incorporate this probability as a new feature of the model.

Fine-grained features A difficulty with the coarse-grained reordering features is that the grammar extraction process finds overlapping rules in the training data and might not give a sensible probability estimate; moreover, reordering statistics from the training data might not carry over perfectly into the translation task (in particular, the training data may have some very freely-reordering translations that one might want to avoid replicating in translation). As an alternative, we introduce a fine-grained version of our distortion model that can be trained directly in the translation task as follows: define a separate binary feature for each value of (R, S), where R is as above and $S \in \{\star, 1, \ldots, 9, \geq 10\}$ and $\star$ means any size. For example, if a nonterminal with span 11 has its contents reordered, then the features (true, $\geq$10) and (true, $\star$) would both fire. Grouping all sizes of 10 or more into a single feature is designed to avoid overfitting.

Again, using MIRA makes it practical to train with the full fine-grained feature set—coincidentally also a total of 34 features.

5 Experiment and results

We now describe our experiments to test MIRA and our features, the soft-syntactic constraints and the structural distortion features, on an Arabic-English translation task. It is worth noting that this experimentation is on a larger scale than Watanabe et al.'s (2007), and considerably larger than Marton and Resnik's (2008).

5.1 Experimental setup

The baseline model was Hiero with the following baseline features (Chiang, 2005; Chiang, 2007):

- two language models
- phrase translation probabilities $p(f \mid e)$ and $p(e \mid f)$
- lexical weighting in both directions (Koehn et al., 2003)
- word penalty
- penalties for:
  - automatically extracted rules
  - identity rules (translating a word into itself)
  - two classes of number/name translation rules
  - glue rules

The probability features are base-100 log-probabilities.

The rules were extracted from all the allowable parallel text from the NIST 2008 evaluation (152+175 million words of Arabic+English), aligned by IBM Model 4 using GIZA++ (union of both directions). Hierarchical rules were extracted from the most in-domain corpora (4.2+5.4 million words) and phrases were extracted from the remainder. We trained the coarse-grained distortion model on 10,000 sentences of the training data.

Two language models were trained, one on data similar to the English side of the parallel text and one on 2 billion words of English. Both were 5-gram models with modified Kneser-Ney smoothing, lossily compressed using a perfect-hashing scheme similar to that of Talbot and Brants (2008) but using minimal perfect hashing (Botelho et al., 2005).

We partitioned the documents of the NIST 2004 (newswire) and 2005 Arabic-English evaluation data into a tuning set (1178 sentences) and a development set (1298 sentences). The test data was the NIST 2006 Arabic-English evaluation data (NIST part, newswire and newsgroups, 1529 sentences).

To obtain syntactic parses for this data, we tokenized it according to the Arabic Treebank standard using AMIRA (Diab et al., 2004), parsed it with the Stanford parser (Klein and Manning, 2003), and then forced the trees back into the MT system's tokenization.[1]

We ran both MERT and MIRA on the tuning set using 20 parallel processors. We stopped MERT when the score on the tuning set stopped increasing, as is common practice, and for MIRA, we used the development set to decide when to stop training.[2] In our runs, MERT took an average of 9 passes through the tuning set and MIRA took an average of 8 passes. (For comparison, Watanabe et al. report decoding their tuning data of 663 sentences 80 times.)

5.2 Results

Table 1 shows the results of our experiments with the training methods and features described above. All significance testing was performed against the first line (MERT baseline) using paired bootstrap resampling (Koehn, 2004).

First of all, we find that MIRA is competitive with MERT when both use the baseline feature set. In-

[1]The only notable consequence this had for our experimentation is that proclitic Arabic prepositions were fused onto the first word of their NP object, so that the PP and NP brackets were coextensive.

[2]We chose this policy for MIRA to avoid overfitting. However, we could have used the tuning set for this purpose, just as with MERT: in none of our runs would this change have made more than a 0.2 BLEU difference on the development set.

			Dev	NIST 06 (NIST part)		
Train	Features	#	nw	nw	ng	nw+ng
MERT	baseline	12	52.0	50.5	32.4	44.6
	syntax (coarse)	14	52.2	50.9	33.0^{+}	45.0^{+}
	syntax (fine)	34	52.1	50.4	33.5^{++}	44.8
	distortion (coarse)	13	52.3	51.3^{+}	34.3^{++}	45.8^{++}
	distortion (fine)	34	52.0	50.9	34.5^{++}	45.5^{++}
MIRA	baseline	12	52.0	49.8^{-}	34.2^{++}	45.3^{++}
	syntax (fine)	34	53.1^{++}	51.3^{+}	34.5^{++}	46.4^{++}
	distortion (fine)	34	53.3^{++}	51.5^{++}	34.7^{++}	46.7^{++}
	distortion+syntax (fine)	56	53.6^{++}	52.0^{++}	35.0^{++}	47.2^{++}

Table 1: Comparison of MERT and MIRA on various feature sets. Key: # = number of features; nw = newswire, ng = newsgroups; + or ++ = significantly better than MERT baseline ($p < 0.05$ or $p < 0.01$, respectively), − = significantly worse than MERT baseline ($p < 0.05$).

deed, the MIRA system scores significantly higher on the test set; but if we break the test set down by genre, we see that the MIRA system does slightly worse on newswire and better on newsgroups. (This is largely attributable to the fact that the MIRA translations tend to be longer than the MERT translations, and the newsgroup references are also relatively longer than the newswire references.)

When we add more features to the model, the two training methods diverge more sharply. When training with MERT, the coarse-grained pair of syntax features yields a small improvement, but the fine-grained syntax features do not yield any further improvement. By contrast, when the fine-grained features are trained using MIRA, they yield substantial improvements. We observe similar behavior for the structural distortion features: MERT is not able to take advantage of the finer-grained features, but MIRA is. Finally, using MIRA to combine both classes of features, 56 in all, produces the largest improvement, 2.6 Bleu points over the MERT baseline on the full test set.

We also tested some of the differences between our training method and Watanabe et al.'s (2007); the results are shown in Table 2. Compared with local updating (line 2), our method of selecting the oracle translation and negative examples does better by 0.5 Bleu points on the development data. Using loss-augmented inference to add negative examples to local updating (line 3) does not appear to help. Nevertheless, the negative examples are important: for if we use our method for selecting the oracle translation without the additional negative examples (line 4), the algorithm fails, generating very long translations and unable to find a weight setting to shorten them. It appears, then, that the additional negative examples enable the algorithm to reliably learn from the enhanced oracle translations.

Setting	Dev
full	53.6
local updating, no LAI	53.1^{-}
local updating, LAI	53.0^{--}
$\mu = 0.5$ oracle, no LAI	failed
no sharing of updates	53.1^{--}

Table 2: Effect of removing various improvements in learning method. Key: − or −− = significantly worse than full system ($p < 0.05$ or $p < 0.01$, respectively); LAI = loss-augmented inference for additional negative examples.

Finally, we compared our parallelization method against a simpler method in which all processors learn independently and their weight vectors are all averaged together (line 5). We see that sharing information among the processors makes a significant difference.

6 Conclusions

In this paper, we have brought together two existing lines of work: the training method of Watanabe et al. (2007), and the models of Chiang (2005) and Marton and Resnik (2008). Watanabe et al.'s work showed that large-margin training with MIRA can be made feasible for state-of-the-art MT systems by using a manageable tuning set; we have demonstrated that parallel processing and exploiting more of the parse forest improves MIRA's performance and that, even using the same set of features, MIRA's performance compares favorably to MERT in terms of both translation quality and computational cost.

Marton and Resnik (2008) showed that it is possible to improve translation in a data-driven framework by incorporating source-side syntactic analysis in the form of soft syntactic constraints. This work joins a growing body of work demonstrating the utility of syntactic information in statistical MT. In the area of source-side syntax, recent research has continued to improve tree-to-string translation models, soften the constraints of the input tree in various ways (Mi et al., 2008; Zhang et al., 2008), and extend phrase-based translation with source-side soft syntactic constraints (Cherry, 2008). All this work shows strong promise, but Marton and Resnik's soft syntactic constraint approach is particularly appealing because it can be used unobtrusively with any hierarchically-structured translation model. Here, we have shown that using MIRA to weight all the constraints at once removes the crucial drawback of the approach, the problem of feature selection.

Finally, we have introduced novel structural distortion features to fill a notable gap in the hierarchical phrase-based approach. By capturing how reordering depends on constituent length, these features improve translation quality significantly. In sum, we have shown that removing the bottleneck of MERT opens the door to many possibilities for better translation.

Acknowledgments

Thanks to Michael Bloodgood for performing initial simulations of parallelized perceptron training. Thanks also to John DeNero, Kevin Knight, Daniel Marcu, and Fei Sha for valuable discussions and suggestions. This research was supported in part by DARPA contract HR0011-06-C-0022 under subcontract to BBN Technologies and HR0011-06-02-001 under subcontract to IBM.

References

Abhishek Arun and Philipp Koehn. 2007. Online learning methods for discriminative training of phrase based statistical machine translation. In *Proc. MT Summit XI*.

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proc. ACL-08: HLT*.

Fabiano C. Botelho, Yoshiharu Kohayakawa, and Nivio Ziviani. 2005. A practical minimal perfect hashing method. In *4th International Workshop on Efficient and Experimental Algorithms (WEA05)*.

Daniel Cer, Daniel Jurafsky, and Christopher D. Manning. 2008. Regularization and search for minimum error rate training. In *Proc. Third Workshop on Statistical Machine Translation*.

Colin Cherry. 2008. Cohesive phrase-based decoding for statistical machine translation. In *Proc. ACL-08: HLT*.

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

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2).

Michael Collins. 2002. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *Proc. EMNLP 2002*.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. 2004. Automatic tagging of Arabic text: From raw text to base phrase chunks. In *Proc. HLT/NAACL 2004*. Companion volume.

Markus Dreyer, Keith Hall, and Sanjeev Khudanpur. 2007. Comparing reordering constraints for SMT using efficient Bleu oracle computation. In *Proc. 2007 Workshop on Syntax and Structure in Statistical Translation*.

Kevin Duh and Katrin Kirchoff. 2008. Beyond log-linear models: Boosted minimum error rate training for n-best re-ranking. In *Proc. ACL-08: HLT, Short Papers*.

Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37:277–296.

Dan Klein and Chris D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15 (NIPS 2002)*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. HLT-NAACL 2003*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. EMNLP 2004*.

Gregor Leusch, Evgeny Matusov, and Hermann Ney. 2008. Complexity of finding the BLEU-optimal hypothesis in a confusion network. In *Proc. EMNLP 2008*. This volume.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proc. COLING-ACL 2006*.

Chin-Yew Lin and Franz Josef Och. 2004. ORANGE: a method for evaluating automatic evaluation metrics for machine translation. In *Proc. COLING 2004*.

Yuval Marton and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrased-based translation. In *Proc. ACL-08: HLT*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. ACL 2005*.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proc. ACL-08: HLT*.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. ACL 2002*.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. ACL 2003*.

John C. Platt. 1998. Fast training of support vector machines using sequential minimal optimization. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, pages 195–208. MIT Press.

David A. Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In *Proc. COLING/ACL 2006, Poster Sessions*.

David Talbot and Thorsten Brants. 2008. Randomized language models via perfect hash functions. In *Proc. ACL-08: HLT*.

Ben Taskar, Dan Klein, Michael Collins, Daphne Koller, and Christopher Manning. 2004. Max-margin parsing. In *Proc. EMNLP 2004*, pages 1–8.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. 2005. Learning structured prediction models: A large margin approach. In *Proc. ICML 2005*.

Christoph Tillmann and Tong Zhang. 2006. A discriminative global training algorithm for statistical MT. In *Proc. COLING-ACL 2006*.

Joseph Turian, Benjamin Wellington, and I. Dan Melamed. 2007. Scalable discriminative learning for natural language parsing and translation. In *Advances in Neural Information Processing Systems 19 (NIPS 2006)*.

Taro Watanabe, Jun Suzuki, Hajime Tsukuda, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proc. EMNLP 2007*.

Dekai Wu. 1996. A polynomial-time algorithm for statistical machine translation. In *Proc. 34th Annual Meeting of the Association for Computational Linguistics*, pages 152–158.

Min Zhang, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan, and Sheng Li. 2008. A tree sequence alignment-based tree-to-tree translation model. In *Proc. ACL-08: HLT*.

A noisy-channel model of rational human sentence comprehension under uncertain input

Roger Levy
Department of Linguistics
University of California – San Diego
9500 Gilman Drive #0108
La Jolla, CA 92093-0108
rlevy@ling.ucsd.edu [*]

Abstract

Language comprehension, as with all other cases of the extraction of meaningful structure from perceptual input, takes places under noisy conditions. If human language comprehension is a rational process in the sense of making use of all available information sources, then we might expect uncertainty at the level of word-level input to affect sentence-level comprehension. However, nearly all contemporary models of sentence comprehension assume *clean* input—that is, that the input to the sentence-level comprehension mechanism is a perfectly-formed, completely certain sequence of input tokens (words). This article presents a simple model of rational human sentence comprehension under noisy input, and uses the model to investigate some outstanding problems in the psycholinguistic literature for theories of rational human sentence comprehension. We argue that by explicitly accounting for input-level noise in sentence processing, our model provides solutions for these outstanding problems and broadens the scope of theories of human sentence comprehension as rational probabilistic inference.

[*] Part of this work has benefited from presentation at the 21st annual meeting of the CUNY Sentence Processing Conference in Chapel Hill, NC, 14 March 2008, and at a seminar at the Center for Research on Language, UC San Diego. I am grateful to Klinton Bicknell, Andy Kehler, and three anonymous reviewers for comments and suggestions, Cyril Allauzen for guidance regarding the OpenFST library, and to Mark Johnson, Mark-Jan Nederhof, and Noah Smith for discussion of renormalizing weighted CFGs.

1 Introduction

Considering the adversity of the conditions under which linguistic communication takes place in everyday life—ambiguity of the signal, environmental competition for our attention, speaker error, and so forth—it is perhaps remarkable that we are as successful at it as we are. Perhaps the leading explanation of this success is that (a) the linguistic signal is redundant, and (b) diverse information sources are generally available that can help us obtain infer the intended message (or something close enough) when comprehending an utterance (Tanenhaus et al., 1995; Altmann and Kamide, 1999; Genzel and Charniak, 2002, 2003; Aylett and Turk, 2004; Keller, 2004; Levy and Jaeger, 2007). Given the difficulty of this task coupled with the availability of redundancy and useful information sources, it would seem rational for all available information to be used to its fullest in sentence comprehension. This idea is either implicit or explicit in several interactivist theories of probabilistic language comprehension (Jurafsky, 1996; Hale, 2001; Narayanan and Jurafsky, 2002; Levy, 2008). However, these theories have implicitly assumed a partitioning of interactivity that distinguishes the *word* as a fundamental level of linguistic information processing: word recognition is an evidential process whose output is nonetheless a specific "winner-takes-all" sequence of words, which is in turn the input to an evidential sentence-comprehension process. It is theoretically possible that this partition is real and is an optimal solution to the problem of language comprehension under gross architectural constraints that favor modularity

(Fodor, 1983). On the other hand, it is also possible that this partition has been a theoretical convenience but that, in fact, evidence at the sub-word level plays an important role in sentence processing, and that sentence-level information can in turn affect word recognition. If the latter is the case, then the question arises of how we might model this type of information flow, and what consequences it might have for our understanding of human language comprehension. This article employs the well-understood formalisms of probabilistic context-free grammars (PCFGs) and weighted finite-state automata (wFSAs) to propose a novel yet simple noisy-channel probabilistic model of sentence comprehension under circumstances where there is uncertainty about word-level representations. Section 2 introduces this model. We use this new model to investigate two outstanding problems for the theory of rational sentence comprehension: one involving *global inference*—the beliefs that a human comprehender arrives at regarding the meaning of a sentence after reading it in its entirety (Section 3)—and one involving *incremental inference*—the beliefs that a comprehender forms and updates moment by moment while reading each part of it (Section 4). The common challenge posed by each of these problems is an apparent failure on the part of the comprehender to use information made available in one part of a sentence to rule out an interpretation of another part of the sentence that is inconsistent with this information. In each case, we will see that the introduction of uncertainty into the input representation, coupled with noisy-channel inference, provides a unified solution within a theory of rational comprehension.

2 Sentence comprehension under uncertain input

The use of generative probabilistic grammars for parsing is well understood (e.g., Charniak, 1997; Collins, 1999). The problem of using a probabilistic grammar G to find the "best parse" T for a known input string $\mathbf{w}$ is formulated as[1]

$$\arg\max_T P_G(T|\mathbf{w}) \qquad \text{(I)}$$

but a *generative* grammar directly defines the joint distribution $P_G(T, \mathbf{w})$ rather than the conditional distribution. In this case, Bayes' rule is used to find the posterior:

$$P_G(T|\mathbf{w}) = \frac{P(T,\mathbf{w})}{P(\mathbf{w})} \qquad \text{(II)}$$

$$\propto P(T,\mathbf{w}) \qquad \text{(III)}$$

If the input string is unknown, the problem changes. Suppose we have some noisy evidence I that determines a probability distribution over input strings $P(\mathbf{w}|I)$. We can still use Bayes' rule to obtain the posterior:

$$P_G(T|I) = \frac{P(T,I)}{P(I)} \qquad \text{(IV)}$$

$$\propto \sum_{\mathbf{w}} P(I|T,\mathbf{w})P(\mathbf{w}|T)P(T) \qquad \text{(V)}$$

Likewise, if we are focused on inferring which words were seen given an uncertain input, we have

$$P_G(\mathbf{w}|I) \propto \sum_T P(I|T,\mathbf{w})P(\mathbf{w}|T)P(T) \qquad \text{(VI)}$$

2.1 Uncertainty for a Known Input

This paper considers situations such as controlled psycholinguistic experiments where we (the researchers) know the sentence $\mathbf{w}^*$ presented to a comprehender, but do not know the specific input I that the comprehender obtains. In this case, if we are, for example, interested in the expected inferences of a rational comprehender about what word string she was exposed to, the probability distribution of interest is

$$P(\mathbf{w}|\mathbf{w}^*) = \int_I P_C(\mathbf{w}|I,\mathbf{w}^*)P_T(I|\mathbf{w}^*)\,dI \quad \text{(VII)}$$

where P_C is the probability distribution used by the comprehender to process perceived input, and P_T is the "true" probability distribution over the inputs

[1]By assumption, G is defined such that its complete productions T completely specify the string, such that $P(\mathbf{w}|T)$ is non-zero for only one value of $\mathbf{w}$.

that might actually be perceived given the true sentence. Since the comprehender does not observe $\mathbf{w}^*$ we must have conditional independence between $\mathbf{w}$ and $\mathbf{w}^*$ given I. We can then apply Bayes' rule to (VII) to obtain

$$P(\mathbf{w}|\mathbf{w}^*) = \int_I \frac{P_C(I|\mathbf{w})P_C(\mathbf{w})}{P_C(I)} P_T(I|\mathbf{w}^*)\, dI \tag{VIII}$$

$$= P_C(\mathbf{w}) \int_I \frac{P_C(I|\mathbf{w})P_T(I|\mathbf{w}^*)}{P_C(I)}\, dI \tag{IX}$$

$$\propto P_C(\mathbf{w})Q(\mathbf{w},\mathbf{w}^*) \tag{X}$$

where $Q(\mathbf{w},\mathbf{w}^*)$ is proportional to the integral term in Equation (IX). The term $P_C(\mathbf{w})$ corresponds to the comprehender's prior beliefs; the integral term is the effect of input uncertainty. If comprehenders model noise rationally, then we should have $P_C(I|\mathbf{w}) = P_T(I|\mathbf{w})$, and thus $Q(\mathbf{w},\mathbf{w}^*)$ becomes a symmetric, non-negative function of $\mathbf{w}$ and $\mathbf{w}^*$; hence the effect of input uncertainty can be modeled by a *kernel function* on input string pairs. (Similar conclusions result when the posterior distribution of interest is over structures T.) It is an open question which kernel functions might best model the inferences made in human sentence comprehension. Most obviously the kernel function should account for noise (environmental, perceptual, and attentional) introduced into the signal en route to the neural stage of abstract sentence processing. In addition, this kernel function might also be a natural means of accounting for modeling error such as disfluencies (Johnson and Charniak, 2004), word/phrase swaps, and even well-formed utterances that the speaker did not intend. For purposes of this paper, we limit ourselves to a simple kernel based on the Levenshtein distance $LD(w,w')$ between words and constructed in the form of a weighted finite-state automaton (Mohri, 1997).

2.2 The Levenshtein-distance kernel

Suppose that the input word string $\mathbf{w}^*$ consists of words $w_{1\ldots n}$. We define the Levenshtein-distance kernel as follows. Start with a weighted finite-state automaton in the log semiring over the vocabulary Σ with states $0\ldots n$, state 0 being the start state

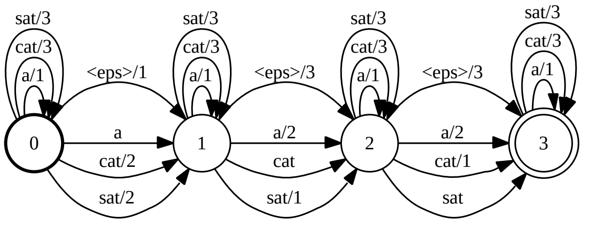


Figure 1: The Levenshtein-distance kernel for multi-word string edits. $K_{LD}(\mathbf{w}^*)$ is shown for $\Sigma = \{\text{cat,sat,a}\}$, $\mathbf{w}^* = (\text{a cat sat})$, and $\lambda = 1$. State 0 is the start state, and State 3 is the lone (zero-cost) final state.

and n the (zero-cost) final state. We add two types of arcs to this automaton: (a) substitution/deletion arcs $(i-1,w') \to i$, $i \in 1,\ldots,n$, each with cost $\lambda\, LD(w_i,w')$, for all $w' \in \Sigma \cup \{\epsilon\}$; and (b) insertion loop arcs $(j,w') \to j$, $j \in 0,\ldots,n$, each with cost $\lambda\, LD(\epsilon,w')$, for all $w' \in \Sigma$.[2] The resulting wFSA $K_{LD}(\mathbf{w}^*)$ defines a function over $\mathbf{w}$ such that the summed weight of paths through the wFSA accepting $\mathbf{w}$ is $\log Q(\mathbf{w},\mathbf{w}^*)$. This kernel allows for the possibility of word *substitutions* (represented by the transition arcs with labels that are neither w_i nor ϵ), word *deletions* (represented by the transition arcs with ϵ labels), and even word *insertions* (represented by the loop arcs). The unnormalized probability of each type of operation is exponential in the Levenshtein distance of the change induced by the operation. The term λ is a free parameter, with smaller values corresponding to noisier input. Figure 1 gives an example of the Levenshtein-distance kernel for a simple vocabulary and sentence.[3]

[2]For purposes of computing the Levenshtein distance between words, the epsilon label ϵ is considered to be a zero-length letter string.

[3]The Levenshtein-distance kernel can be seen to be symmetric in $\mathbf{w},\mathbf{w}^*$ as follows. Any path accepting $\mathbf{w}$ in the wFSA generated from $\mathbf{w}^*$ involves the following non-zero-cost transitions: insertions $w'^{I}_{1\ldots i}$, deletions $w^{D}_{1\ldots j}$, and substitutions $(w \to w')^{S}_{1\ldots k}$. For each such path P, there will be exactly one path P' in the wFSA generated from $\mathbf{w}$ that accepts $\mathbf{w}^*$ with insertions $w^{D}_{1\ldots j}$, deletions $w'^{I}_{1\ldots i}$, and substitutions $(w' \to w)^{S}_{1\ldots k}$. Due to the symmetry of the Levenshtein distance, the paths P and P' will have identical costs. Therefore the kernel is indeed symmetric.

2.3 Efficient computation of posterior beliefs

The problem of finding structures or strings with high posterior probability given a particular input string $\mathbf{w}^*$ is quite similar to the problem faced in the parsing of speech, where the acoustic input I to a parser can be represented as a lattice of possible word sequences, and the edges of the lattice have weights determined by a model of acoustic realization of words, $P(I|\mathbf{w})$ (Collins et al., 2004; Hall and Johnson, 2003, 2004). The two major differences between lattice parsing and our problem are (a) we have integrated out the expected effect of noise, which is thus implicit in our choice of kernel; and (b) the loops in the Levenshtein-distance kernel mean that the input to parsing is no longer a lattice. This latter difference means that some of the techniques applicable to string parsing and lattice parsing – notably the computation of inside probabilities – are no longer possible using exact methods. We return to this difference in Sections 3 and 4.

3 Global inference

One clear prediction of the uncertain-input model of (VII)–(X) is that under appropriate circumstances, the prior expectations $P_C(\mathbf{w})$ of the comprehender should in principle be able to override the linguistic input actually presented, so that a sentence is interpreted as meaning—and perhaps even *being*—something other than it actually meant or was. At one level, it is totally clear that comprehenders do this on a regular basis: the ability to do this is required for someone to act as a copy editor—that is, to notice and (crucially) correct mistakes on the printed page. In many cases, these types of correction happen at a level that may be below consciousness—thus we sometimes miss a typo but interpret the sentences as it was intended, or ignore the disfluency of a speaker. What has not been previously proposed in a formal model, however, is that this can happen *even when an input is a completely grammatical sentence*. Here, we argue that an effect demonstrated by Christianson et al. (2001) (see also Ferreira et al., 2002) is an example of expectations overriding input. When presented sentences of the forms in (1) using methods that did not permit rereading, and asked questions of the type *Did the man hunt the deer?*, experimental participants gave affirmative responses significantly more often for sentences of type (1a), in which the substring *the man hunted the deer* appears, than for either (1b) or (1c).

(1) a. While the man hunted the deer ran into the woods. (GARDENPATH)
    b. While the man hunted the pheasant the deer ran into the woods. (TRANSITIVE)
    c. While the man hunted, the deer ran into the woods. (COMMA)

This result was interpreted by Christianson et al. (2001) and Ferreira et al. (2002) as reflecting (i) the fact that there is a syntactic garden path in (1a)—after reading the first six words of the sentence, the preferred interpretation of the substring *the man hunted the deer* is as a simple clause indicating that the deer was hunted by the man—and (ii) that readers were not always successful at revising away this interpretation when they saw the disambiguating verb *ran*, which signals that *the deer* is actually the subject of the main clause, and that *hunted* must therefore be intransitive. Furthermore (and crucially), for (1a) participants also responded affirmatively most of the time to questions of the type *Did the deer run into the woods?* This result is a puzzle for existing models of sentence comprehension because no grammatical analysis exists of any substring of (1a) for which *the deer* is both the object of *hunted* and the subject of *ran*. In fact, no formal model has yet been proposed to account for this effect.

The uncertain-input model gives us a means of accounting for these results, because there are near neighbors of (1a) for which there *is* a global grammatical analysis in which either *the deer* or a coreferent NP is in fact the object of the subordinate-clause verb *hunted*. In particular, inserting the word *it* either before or after *the deer* creates such a near neighbor:

(2) a. While the man hunted the deer it ran into the woods.
    b. While the man hunted it the deer ran into the woods.

We formalize this intuition within our model by using the wFSA representation of the Levenshtein-

ROOT	$\rightarrow$ S PUNCT.	0.0
S	$\rightarrow$ SBAR S	6.3
S	$\rightarrow$ SBAR PUNCT_S	4.6
PUNCT_S	$\rightarrow$, S	0.0
S	$\rightarrow$ NP VP	0.1
SBAR	$\rightarrow$ IN S	0.0
NP	$\rightarrow$ DT NN	1.9
NP	$\rightarrow$ NNS	4.4
NP	$\rightarrow$ NNP	3.3
NP	$\rightarrow$ DT NNS	4.5
NP	$\rightarrow$ PRP	1.3
NP	$\rightarrow$ NN	3.1
VP	$\rightarrow$ VBD RB	9.7
VP	$\rightarrow$ VBD PP	2.2
VP	$\rightarrow$ VBD NP	1.2
VP	$\rightarrow$ VBD RP	8.3
VP	$\rightarrow$ VBD	2.0
VP	$\rightarrow$ VBD JJ	3.4
PP	$\rightarrow$ IN NP	0.0

Figure 2: The PCFG used in the global-inference study of Section 3. Rule weights given as negative log-probabilities in bits.

distance kernel. A probabilistic context-free grammar (PCFG) representing the comprehender's grammatical knowledge can be intersected with that wFSA using well-understood techniques, generating a new weighted CFG (Bar-Hillel et al., 1964; Nederhof and Satta, 2003). This intersection thus represents the unnormalized posterior $P_C(T, \mathbf{w}|\mathbf{w}^*)$. Because there are loops in the wFSA generated by the Levenshtein-distance kernel, exact normalization of the posterior is not tractable (though see Nederhof and Satta, 2003; Chi, 1999; Smith and Johnson, 2007 for possible approaches to approximating the normalization constant). We can, however, use the lazy k-best algorithm of Huang and Chiang (2005; Algorithm 3) to obtain the word-string/parse-tree pairs with highest posterior probability.

3.1 Experimental Verification

To test our account of the rational noisy-channel interpretation of sentences such as (1), we defined a small PCFG using the phrasal rules listed in Figure 2, with rule probabilities estimated from the parsed Brown corpus.[4] Lexical rewrite probabilities were determined using relative-frequency estimation over the entire parsed Brown corpus. For each of the sentence sets like (1) used in Experiments 1a, 1b, and 2 of Christianson et al. (2001) that have complete lexical coverage in the parsed Brown corpus (22 sets in total), a noisy-input wFSA was constructed using K_{LD}, permitting all words occurring more than 2500 times in the parsed Brown corpus as possible edit/insertion targets.[5] Figure 3 shows the average proportion of parse trees among the 100 best parses in the intersection between this PCFG and the wFSA for each sentence for which an interpretation is available such that *the deer* or a coreferent NP is the direct object of *hunted*.[6] The Levenshtein distance penalty λ is a free parameter in the model, but the results are consistent for a wide range of λ: interpretations of type (2) are more prevalent both in terms of number mass for (1a) than for either (1b) or (1c). Furthermore, across 9 noise values for 22 sentence sets, there were never more interpretations of type (2) for COMMA sentences than for the corresponding GARDENPATH sentences, and in only one case were there more such interpretations for a TRANSITIVE sentence than for the corresponding GARDENPATH sentence.

4 Incremental comprehension and error identification

We begin taking up the role of input uncertainty for incremental comprehension by posing a question:

[4]Counts of these rules were obtained using `tgrep2/Tregex` tree-matching patterns (Rohde, 2005; Levy and Andrew, 2006), available online at `http://idiom.ucsd.edu/~rlevy/papers/emnlp2008/tregex_patterns`. We have also investigated the use of broad-coverage PCFGs estimated using standard treebank-based techniques, but found that the computational cost of inference with treebank-sized grammars was prohibitive.

[5]The word-frequency cutoff was introduced for computational speed; we have obtained qualitatively similar results with lower word-frequency cutoffs.

[6]We took a parse tree to satisfy this criterion if the NP *the deer* appeared either as the matrix-clause subject or the embedded-clause object, and a pronoun appeared in the other position. In a finer-grained grammatical model, number/gender agreement would be enforced between such a pronoun and the NP in the posterior, so that the probability mass for these parses would be concentrated on cases where the pronoun is *it*.

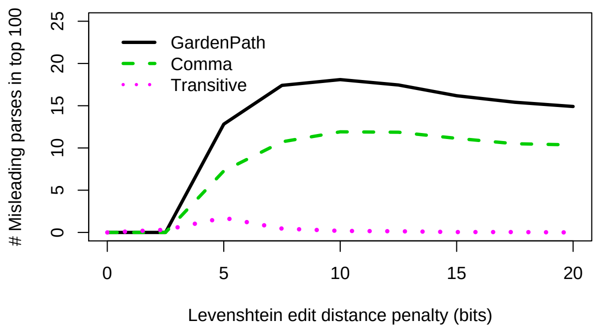


Figure 3: Results for 100-best global inference, as a function of the Levenshtein distance penalty λ (in bits).

what is the optimal way to read a sentence on a page (Legge et al., 1997)? Presumably, the goal of reading is to find a good compromise between scanning the contents of the sentence as quickly as possible while achieving an accurate understanding of the sentence's meaning. To a first approximation, humans solve this problem by reading each sentence in a document from beginning to end, regardless of the actual layout; whether this general solution is best understood in terms of optimality or rather as parasitic on spoken language comprehension is an open question beyond the immediate scope of the present paper. However, about 10–15% of eye movements in reading are regressive (Rayner, 1998), and we may usefully refine our question to when a *regressive* eye movement might be a good decision. In traditional models of sentence comprehension, the optimal answer would certainly be "never", since past observations are known with certainty. But once uncertainty about the past is accounted for, it is clear that there may in principle be situations in which regressive saccades may be the best choice.

What are these situations? One possible answer would be: when the uncertainty (e.g., measured by entropy) about an earlier part of the sentence is high. There are some cases in which this is probably the correct answer: many regressive eye movements are very small and the consensus in the eye-movement literature is that they represent corrections for motor error at the saccadic level. That is, the eyes overshoot the intended target and regress to obtain information about what was missed. However, motor error can account only for short, isolated regressions, and about one-sixth of regressions are part of a longer series back into the sentence, into a much earlier part of the text which has already been read. We propose that these regressive saccades might be the best choice *when the most recent observed input significantly changes the comprehender's beliefs about the earlier parts of the sentence*. To make the discussion more concrete, we turn to another recent result in the psycholinguistic literature that has been argued to be problematic for rational theories of sentence comprehension.

It has been shown (Tabor et al., 2004) that sentences such as (3) below induce considerable processing difficulty at the word *tossed*, as measured in word-by-word reading times:

(3) The coach smiled at the player tossed a frisbee. (LOCALLY COHERENT)

Both intuition and controlled experiments reveal that this difficulty seems due at least in part to the category ambiguity of the word *tossed*, which is occasionally used as a participial verb but is much more frequently used as a simple-past verb. Although *tossed* in (3) is actually a participial verb introducing a reduced relative clause (and *the player* is hence its recipient), most native English speakers find it extremely difficult not to interpret *tossed* as a main verb and *the player* as its agent—far more difficult than for corresponding sentences in which the critical participial verb is morphologically distinct from the simple past form ((4a), (4c); c.f. *threw*) or in which the relative clause is unreduced and thus clearly marked ((4b), (4c)).

(4) a. The coach smiled at the player thrown a frisbee. (LOCALLY INCOHERENT)
b. The coach smiled at the player who was tossed a frisbee.
c. The coach smiled at the player who was thrown a frisbee.

The puzzle here for rational approaches to sentence comprehension is that the preceding top-down context provided by *The coach smiled at...* should completely rule out the possibility of seeing a main verb immediately after *player*, hence a rational comprehender should not be distracted by the part-of-speech ambiguity.[7]

4.1 An uncertain-input solution

The solution we pursue to this puzzle lies in the fact that (3) has many near-neighbor sentences in which the word *tossed* is in fact a simple-past tense verb. Several possibilities are listed below in (5):

(5)
a. The coach **who** smiled at the player tossed a frisbee.
b. The coach smiled **as** the player tossed a frisbee.
c. The coach smiled **and** the player tossed a frisbee.
d. The coach smiled at the player **who** tossed a frisbee.
e. The coach smiled at the player **that** tossed a frisbee.
f. The coach smiled at the player **and** tossed a frisbee.

The basic intuition we follow is that simple-past verb *tossed* is much more probable where it appears in any of (5a)-(5f) than participial *tossed* is in (3). Therefore, seeing this word causes the comprehender to shift her probability distribution about the earlier part of the sentence away from (3), where it had been peaked, toward its near neighbors such as the examples in (5). This change in beliefs about the past is treated as an error identification signal (EIS). In reading, a sensible response to an EIS would be a slowdown or a regressive saccade; in spoken language comprehension, a sensible response would be to allocate more working memory resources to the comprehension task.

4.2 Quantifying the Error Identification Signal

We quantify our proposed error identification signal as follows. Consider the probability distribution over the input up to, but not including, a position j in a sentence $\mathbf{w}$:

7This preceding context sharply distinguishes (3) from better-known, traditional garden-path sentences such as *The horse raced past the barn fell*, in which preceding context cannot be used to correctly disambiguate the part of speech of the ambiguous verb *raced*.

$$P(w_{[0,j)}) \qquad \text{(XI)}$$

We use the subscripting $[0, j)$ to illustrate that this interval is “closed” through to include the beginning of the string, but “open” at position j—that is, it includes all material before position j but does not include anything at that position or beyond. Let us then define the posterior distribution after seeing all input up through and including word i as $P_i(w_{[0,j)})$. We define the EIS induced by reading a word w_i as follows:

$$D\left(P_i(w_{[0,i)})||P_{i-1}(w_{[0,i)})\right) \qquad \text{(XII)}$$

$$\equiv \sum_{w \in \{w_{[0,i)}\}} P_i(w) \log \frac{P_i(w)}{P_{i-1}(w)} \qquad \text{(XIII)}$$

where $D(q||p)$ is the Kullback-Leibler divergence, or relative entropy, from p to q, a natural way of quantifying the distance between probability distributions (Cover and Thomas, 1991) which has also been argued for previously in modeling attention and surprise in both visual and linguistic cognition (Itti and Baldi, 2005; Levy, 2008).

4.3 Experimental Verification

As in Section 3, we use a small probabilistic grammar covering the relevant structures in the problem domain to represent the comprehender's knowledge, and a wFSA based on the Levenshtein-distance kernel to represent noisy input. We are interested in comparing the EIS at the word *tossed* in (3) versus the EIS at the word *thrown* in (4a). In this case, the interval $w_{[0,j)}$ contains all the material that could possibly have come before the word *tossed/thrown*, but does not contain material at or after the position introduced by the word itself. Loops in the probabilistic grammar and the Levenshtein-distance kernel pose a challenge, however, to evaluating the EIS, because the normalization constant of the resulting grammar/input intersection is essential to evaluating Equation (XIII). To circumvent this problem, we eliminate loops from the kernel by allowing only one insertion per inter-word space.[8] (See Section 5 for a possible alternative).

[8]Technically, this involves the following transformation of a Levenshtein-distance wFSA. First, eliminate all loop arcs.

ROOT	$\rightarrow$ S	0.00
S	$\rightarrow$ S-base CC S-base	7.3
S	$\rightarrow$ S-base	0.01
S-base	$\rightarrow$ NP-base VP	0
NP	$\rightarrow$ NP-base RC	4.1
NP	$\rightarrow$ NP-base	0.5
NP	$\rightarrow$ NP-base PP	2.0
NP-base	$\rightarrow$ DT N N	4.7
NP-base	$\rightarrow$ DT N	1.9
NP-base	$\rightarrow$ DT JJ N	3.8
NP-base	$\rightarrow$ PRP	1.0
NP-base	$\rightarrow$ NNP	3.1
VP/NP	$\rightarrow$ V NP	4.0
VP/NP	$\rightarrow$ V	0.1
VP	$\rightarrow$ V PP	2.0
VP	$\rightarrow$ V NP	0.7
VP	$\rightarrow$ V	2.9
RC	$\rightarrow$ WP S/NP	0.5
RC	$\rightarrow$ VP-pass/NP	2.0
RC	$\rightarrow$ WP FinCop VP-pass/NP	4.9
PP	$\rightarrow$ IN NP	0
S/NP	$\rightarrow$ VP	0.7
S/NP	$\rightarrow$ NP-base VP/NP	1.3
VP-pass/NP	$\rightarrow$ VBN NP	2.2
VP-pass/NP	$\rightarrow$ VBN	0.4

Figure 4: The grammar used for the incremental-inference experiment of Section 4. Rule weights given as negative log-probabilities in bits.

Figure 4 shows the (finite-state) probabilistic grammar used for the study, with rule probabilities once again determined from the parsed Brown corpus using relative frequency estimation. To calculate the distribution over strings after exposure to the i-th word in the sentence, we "cut" the input wFSA such that all transitions and arcs after state $2i+2$ were removed and replaced with a sequence of states $j = 2i+3, \ldots, m$, with zero-cost transitions $(j-1, w') \rightarrow j$ for all $w' \in \Sigma \cup \{\epsilon\}$, and each new j being a zero-cost final state.[9] Because the intersection between this "cut" wFSA and the probabilistic grammar is loop-free, it can be renormalized, and the EIS can be calculated without difficulty. All the computations in this section were carried out using the OpenFST library (Allauzen et al., 2007).

Next, map every state i onto a state pair in a new wFSA $(2i, 2i+1)$, with all incoming arcs in i being incoming into $2i$, all outgoing arcs from i being outgoing from $2i+1$, and new transition arcs $(2i, w') \rightarrow 2i+1$ for each $w' \in \Sigma \cup \{\epsilon\}$ with cost $LD(\epsilon, w')$. Finally, add initial state 0 to the new wFSA with transition arcs to state 1 for all $w' \in \Sigma \cup \{\epsilon\}$ with cost $LD(\epsilon, w')$. A final state i in the old wFSA corresponds to a final state $2i+1$ in the new wFSA.

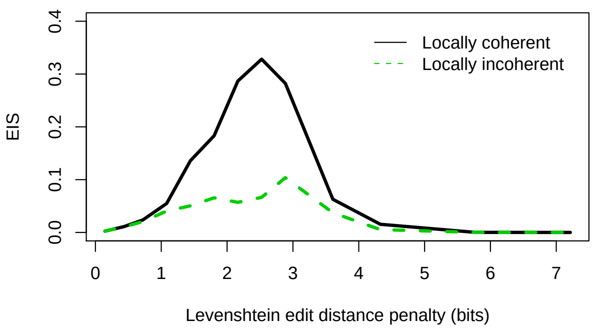


Figure 5: The Error Identification Signal (EIS) for (3) and (4a), as a function of the Levenshtein distance penalty λ (in bits)

Figure 5 shows the average magnitude of the EIS for sentences (3) versus (4a) at the critical word position *tossed/thrown*. Once again, the Levenshtein-distance penalty λ is a free parameter in the model, so we show model behavior as a function of λ, for the eight sentence pairs in Experiment 1 of Tabor et al. with complete lexical and syntactic coverage for the grammar of Figure 4. For values of λ where the EIS is non-negligible, it is consistently larger at the critical word (*tossed* in (3), *thrown* in (4a)) in the COHERENT condition than in the INCOHERENT condition. Across a range of eight noise levels, 67% of sentence pairs had a higher EIS in the COHERENT condition than in the INCOHERENT condition. Furthermore, the cases where the INCOHERENT condition had a larger EIS occurred only for noise levels below 1.1 and above 3.6, and the maximum such EIS was quite small, at 0.067. Overall, the model's behavior is consistent with the experimental results of Tabor et al. (2004), and can be explained through the intuition described at the end of Section 4.1.

[9]The number of states added had little effect on results, so long as at least as many states were added as words remained in the sentence.

5 Conclusion

In this paper we have outlined a simple model of rational sentence comprehension under uncertain input and explored some of the consequences for outstanding problems in the psycholinguistic literature. The model proposed here will require further empirical investigation in order to distinguish it from other proposals that have been made in the literature, but if our proposal turns out to be correct it has important consequences for both the theory of language processing and cognition more generally. Most notably, it furthers the case for rationality in sentence processing; and it eliminates one of the longest-standing modularity hypotheses implicit in work on the cognitive science of language: a partition between systems of word recognition and sentence comprehension (Fodor, 1983). Unlike the pessimistic picture originally painted by Fodor, however, the interactivist picture resulting from our model's joint inference over possible word strings and structures points to many rich details that still need to be filled in. These include questions such as what kernel functions best account for human comprehenders' modeling of noise in linguistic input, and what kinds of algorithms might allow representations with uncertain input to be computed incrementally.

The present work could also be extended in several more technical directions. Perhaps most notable is the problem of the normalization constant for the posterior distribution over word strings and structures; this problem was circumvented via a k-best approach in Section 3 and by removing loops from the Levenshtein-distance kernel in Section 4. We believe, however, that a more satisfactory solution may exist via sampling from the posterior distribution over trees and strings. This may be possible either by estimating normalizing constants for the posterior grammar using iterative weight propagation and using them to obtain proper production rule probabilities (Chi, 1999; Smith and Johnson, 2007), or by using reversible-jump Markov-chain Monte Carlo (MCMC) techniques to sample from the posterior (Green, 1995), and estimating the normalizing constant with annealing-based techniques (Gelman and Meng, 1998) or nested sampling (Skilling, 2004). Scaling the model up for use with treebank-size grammars is another area for technical improvement.

Finally, we note that the model here could potentially find practical application in grammar correction. Although the noisy channel has been in use for many years in spelling correction, our model could be used more generally for grammar corrections, including insertions, deletions, and (with new noise functions) potentially changes in word order.

References

Allauzen, C., Riley, M., Schalkwyk, J., Skut, W., and Mohri, M. (2007). OpenFst: A general and efficient weighted finite-state transducer library. In *Proceedings of the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007)*, volume 4783 of *Lecture Notes in Computer Science*, pages 11–23. Springer. `http://www.openfst.org`.

Altmann, G. T. and Kamide, Y. (1999). Incremental interpretation at verbs: restricting the domain of subsequent reference. *Cognition*, 73(3):247–264.

Aylett, M. and Turk, A. (2004). The Smooth Signal Redundancy Hypothesis: A functional explanation for relationships between redundancy, prosodic prominence, and duration in spontaneous speech. *Language and Speech*, 47(1):31–56.

Bar-Hillel, Y., Perles, M., and Shamir, E. (1964). On formal properties of simple phrase structure grammars. In *Language and Information: Selected Essays on their Theory and Application*. Addison-Wesley.

Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. In *Proceedings of AAAI*, pages 598–603.

Chi, Z. (1999). Statistical properties of probabilistic context-free grammars. *Computational Linguistics*, 25(1):131–160.

Christianson, K., Hollingworth, A., Halliwell, J. F., and Ferreira, F. (2001). Thematic roles assigned along the garden path linger. *Cognitive Psychology*, 42:368–407.

Collins, C., Carpenter, B., and Penn, G. (2004). Head-driven parsing for word lattices. In *Proceedings of ACL*.

Collins, M. (1999). *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania.

Cover, T. and Thomas, J. (1991). *Elements of Information Theory*. John Wiley.

Ferreira, F., Ferraro, V., and Bailey, K. G. D. (2002). Good-enough representations in language comprehension. *Current Directions in Psychological Science*, 11:11–15.

Fodor, J. A. (1983). *The Modularity of Mind*. MIT Press.

Gelman, A. and Meng, X.-L. (1998). Simulating normalizing constants: from importance sampling to bridge sampling to path sampling. *Statistical Science*, 13(2):163–185.

Genzel, D. and Charniak, E. (2002). Entropy rate constancy in text. In *Proceedings of ACL*.

Genzel, D. and Charniak, E. (2003). Variation of entropy and parse trees of sentences as a function of the sentence number. In *Empirical Methods in Natural Language Processing*, volume 10.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo and Bayesian model determination. *Biometrika*, 82:711–732.

Hale, J. (2001). A probabilistic Earley parser as a psycholinguistic model. In *Proceedings of NAACL*, volume 2, pages 159–166.

Hall, K. and Johnson, M. (2003). Language modeling using efficient best-first bottom-up parsing. In *Proceedings of the IEEE workshop on Automatic Speech Recognition and Understanding*.

Hall, K. and Johnson, M. (2004). Attention shifting for parsing speech. In *Proceedings of ACL*.

Huang, L. and Chiang, D. (2005). Better k-best parsing. In *Proceedings of the International Workshop on Parsing Technologies*.

Itti, L. and Baldi, P. (2005). Bayesian surprise attracts human attention. In *Advances in Neural Information Processing Systems*.

Johnson, M. and Charniak, E. (2004). A TAG-based noisy channel model of speech repairs. In *Proceedings of ACL*.

Jurafsky, D. (1996). A probabilistic model of lexical and syntactic access and disambiguation. *Cognitive Science*, 20(2):137–194.

Keller, F. (2004). The entropy rate principle as a predictor of processing effort: An evaluation against eye-tracking data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 317–324, Barcelona.

Legge, G. E., Klitz, T. S., and Tjan, B. S. (1997). Mr. Chips: An ideal-observer model of reading. *Psychological Review*, 104(3):524–553.

Levy, R. (2008). Expectation-based syntactic comprehension. *Cognition*, 106:1126–1177.

Levy, R. and Andrew, G. (2006). Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In *Proceedings of the 2006 conference on Language Resources and Evaluation*.

Levy, R. and Jaeger, T. F. (2007). Speakers optimize information density through syntactic reduction. In *Advances in Neural Information Processing Systems*.

Mohri, M. (1997). Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311.

Narayanan, S. and Jurafsky, D. (2002). A Bayesian model predicts human parse preference and reading time in sentence processing. In *Advances in Neural Information Processing Systems*, volume 14, pages 59–65.

Nederhof, M.-J. and Satta, G. (2003). Probabilistic parsing as intersection. In *Proceedings of the International Workshop on Parsing Technologies*.

Rayner, K. (1998). Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin*, 124(3):372–422.

Rohde, D. (2005). *TGrep2 User Manual*, version 1.15 edition.

Skilling, J. (2004). Nested sampling. In Fischer, R., Preuss, R., and von Toussaint, U., editors, *Bayesian inference and maximum entropy methods in science and engineering*, number 735 in AIP Conference Proceedings, pages 395–405.

Smith, N. A. and Johnson, M. (2007). Weighted and probabilistic context-free grammars are equally expressive. *Computational Linguistics*, 33(4):477–491.

Tabor, W., Galantucci, B., and Richardson, D. (2004). Effects of merely local syntactic coherence on sentence processing. *Journal of Memory and Language*, 50(4):355–370.

Tanenhaus, M. K., Spivey-Knowlton, M. J., Eberhard, K., and Sedivy, J. C. (1995). Integration of visual and linguistic information in spoken language comprehension. *Science*, 268:1632–1634.

Incorporating Temporal and Semantic Information with Eye Gaze for Automatic Word Acquisition in Multimodal Conversational Systems

Shaolin Qu **Joyce Y. Chai**
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824
{qushaoli,jchai}@cse.msu.edu

Abstract

One major bottleneck in conversational systems is their incapability in interpreting unexpected user language inputs such as out-of-vocabulary words. To overcome this problem, conversational systems must be able to learn new words automatically during human machine conversation. Motivated by psycholinguistic findings on eye gaze and human language processing, we are developing techniques to incorporate human eye gaze for automatic word acquisition in multimodal conversational systems. This paper investigates the use of temporal alignment between speech and eye gaze and the use of domain knowledge in word acquisition. Our experiment results indicate that eye gaze provides a potential channel for automatically acquiring new words. The use of extra temporal and domain knowledge can significantly improve acquisition performance.

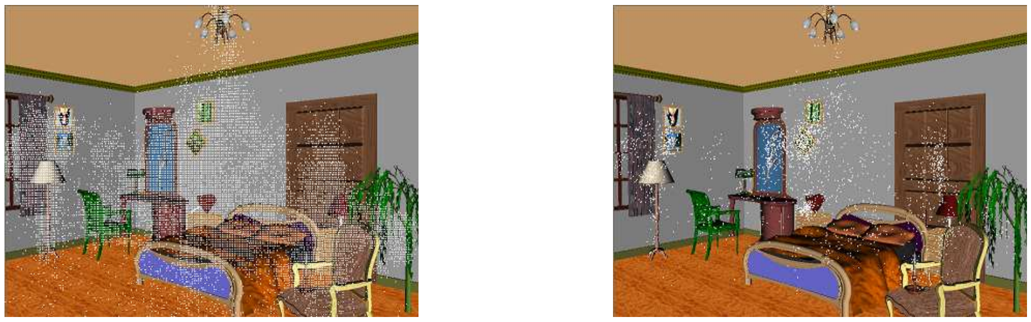
1 Introduction

Interpreting human language is a challenging problem in human machine conversational systems due to the flexibility of human language behavior. When the encountered vocabulary is outside of the system's knowledge, conversational systems tend to fail. It is desirable that conversational systems can learn new words automatically during human machine conversation. While automatic word acquisition in general is quite challenging, multimodal conversational systems offer an unique opportunity to explore word acquisition. In a multimodal conversational system where users can talk and interact with a graphical display, users' eye gaze, which occurs naturally with speech production, provides a potential channel for the system to learn new words automatically during human machine conversation.

Psycholinguistic studies 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 eye gaze carries information about the focus of the user's attention (Just and Carpenter, 1976). The perceived visual context influences spoken word recognition and mediates syntactic processing of spoken sentences (Tanenhaus et al., 1995). 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 investigating the use of eye gaze for automatic word acquisition in multimodal conversation. Particulary, this paper investigates the use of temporal information about speech and eye gaze and domain semantic relatedness for automatic word acquisition. The domain semantic and temporal information are incorporated in statistical translation models for word acquisition. Our experiments show that the use of domain semantic and temporal information significantly improves word acquisition performance.

In the following sections, we first describe the basic translation models for word acquisition. Then, we describe the enhanced models that incorporate temporal and semantic information about speech and eye gaze for word acquisition. Finally, we present the results of empirical evaluation.



(a) Raw gaze points (b) Processed gaze fixations

Figure 1: Domain scene with a user's gaze fixations

2 Related Work

Word acquisition by grounding words to visual entities has been studied in many language grounding systems. For example, given speech paired with video images of single objects, mutual information between audio and visual signals was used to acquire words by associating acoustic phone sequences with the visual prototypes (e.g., color, size, shape) of objects (Roy and Pentland, 2002). Generative models were used to acquire words by associating words with image regions given parallel data of pictures and description text (Barnard et al., 2003). Different from these works, in our work, the visual attention foci accompanying speech are indicated by eye gaze. Eye gaze is an implicit and subconscious input, which brings additional challenges in word acquisition.

Eye gaze has been explored for word acquisition in previous work. In (Yu and Ballard, 2004), given speech paired with eye gaze information and video images, a translation model was used to acquire words by associating acoustic phone sequences with visual representations of objects and actions. A recent investigation on word acquisition from transcribed speech and eye gaze in human machine conversation was reported in (Liu et al., 2007). In this work, a translation model was developed to associate words with visual objects on a graphical display. Different from these previous works, here we investigate the incorporation of extra knowledge, specifically speech-gaze temporal information and domain knowledge, with eye gaze to facilitate word acquisition.

3 Data Collection

We recruited users to interact with a simplified multimodal conversational system to collect speech and eye gaze data.

3.1 Domain

We are working on a 3D room decoration domain. Figure 1 shows the 3D room scene that was shown to the user in the experiments. There are 28 3D objects (bed, chairs, paintings, lamp, etc.) in the room scene. During the human machine conversation, the system verbally asked the user a question (e.g., "*what do you dislike about the arrangement of the room*?") or issued a request (e.g., "*describe the left wall*") about the room. The user provided responses by speaking to the system.

During the experiments, users' speech was recorded through an open microphone and users' eye gaze was captured by an Eye Link II eye tracker. Eye gaze data consists of the screen coordinates of each gaze point that was captured by the eye tracker at a sampling rate of 250hz.

3.2 Data Preprocessing

As for speech data, we collected 357 spoken utterances from 7 users' experiments. The vocabulary size is 480, among which 227 words are nouns and adjectives. We manually transcribed the collected speech.

As for gaze data, the first step is to identify gaze fixation from raw gaze points. As shown in Figure 1(a), the collected raw gaze points are very noisy. They can not be used directly for identifying gaze fixated entities in the scene. We processed the raw gaze data to eliminate invalid and saccadic gaze points. Invalid gaze points occur when users look off the screen. Saccadic gaze points occur during ballistic eye movements between gaze fixations. Vision studies have shown that no visual processing occurs in the human mind during saccades (i.e., saccadic suppression) (Matin, 1974). Since eyes do not stay still but rather make small, frequent jerky movements, we average nearby gaze points to better identify gaze fixations. The processed eye gaze fixations are shown in Figure 1(b).

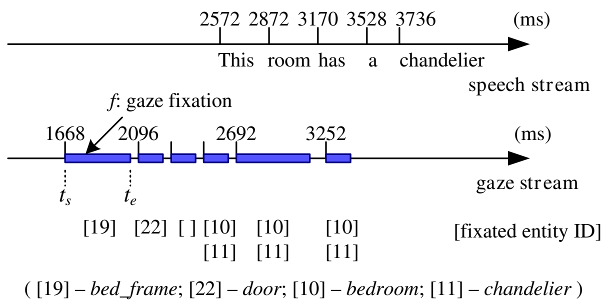


Figure 2: Parallel speech and gaze streams

Figure 2 shows an excerpt of the collected speech and gaze fixation in one experiment. In the speech stream, each word starts at a particular timestamp. In the gaze stream, each gaze fixation has a starting timestamp t_s and an ending timestamp t_e. Each gaze fixation also has a list of fixated entities (3D objects). An entity e on the graphical display is fixated by gaze fixation f if the area of e contains fixation point of f.

Given the collected speech and gaze fixations, we build parallel speech-gaze data set as follows. For each spoken utterance and its accompanying gaze fixations, we construct a pair of word sequence and entity sequence $(\mathbf{w}, \mathbf{e})$. The word sequence $\mathbf{w}$ consists of only nouns and adjectives in the utterance. Each gaze fixation results in a fixated entity in the entity sequence $\mathbf{e}$. When multiple entities are fixated by one gaze fixation due to the overlapping of the entities, the forefront one is chosen. Also, we merge the neighboring gaze fixations that contain the same fixated entities. For the parallel speech and gaze streams shown in Figure 2, the resulting word sequence is $\mathbf{w}$ = [room chandelier] and the entity sequence is $\mathbf{e}$ = [*bed_frame door chandelier*].

4 Translation Models for Automatic Word Acquisition

Since we are working on conversational systems where users interact with a visual scene, we consider the task of word acquisition as associating words with visual entities in the domain. Given the parallel speech and gaze fixated entities $\{(\mathbf{w}, \mathbf{e})\}$, we formulate word acquisition as a translation problem and use translation models to estimate word-entity association probabilities $p(w|e)$. The words with the highest association probabilities are chosen as acquired words for entity e.

4.1 Base Model I

Using the translation model I (Brown et al., 1993), where each word is equally likely to be aligned with each entity, we have

$$p(\mathbf{w}|\mathbf{e}) = \frac{1}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} p(w_j|e_i) \qquad (1)$$

where l and m are the lengths of entity and word sequences respectively. This is the model used in (Liu et al., 2007) and (Yu and Ballard, 2004). We refer to this model as **Model-1** throughout the rest of this paper.

4.2 Base Model II

Using the translation model II (Brown et al., 1993), where alignments are dependent on word/entity positions and word/entity sequence lengths, we have

$$p(\mathbf{w}|\mathbf{e}) = \prod_{j=1}^{m} \sum_{i=0}^{l} p(a_j = i|j, m, l) p(w_j|e_i) \qquad (2)$$

where $a_j = i$ means that w_j is aligned with e_i. When $a_j = 0$, w_j is not aligned with any entity (e_0 represents a *null* entity). We refer to this model as **Model-2**.

Compared to Model-1, Model-2 considers the ordering of words and entities in word acquisition. EM algorithms are used to estimate the probabilities $p(w|e)$ in the translation models.

5 Using Speech-Gaze Temporal Information for Word Acquisition

In Model-2, word-entity alignments are estimated from co-occurring word and entity sequences in an unsupervised way. The estimated alignments are dependent on where the words/entities appear in the word/entity sequences, not on when those words and gaze fixated entities actually occur. Motivated by the finding that users move their eyes to the mentioned object directly before speaking a word (Griffin and Bock, 2000), we make the word-entity alignments dependent on their temporal relation in a new model (referred as **Model-2t**):

$$p(\mathbf{w}|\mathbf{e}) = \prod_{j=1}^{m} \sum_{i=0}^{l} p_t(a_j = i|j, \mathbf{e}, \mathbf{w}) p(w_j|e_i) \quad (3)$$

where $p_t(a_j = i|j, \mathbf{e}, \mathbf{w})$ is the temporal alignment probability computed based on the temporal distance between entity e_i and word w_j.

We define the temporal distance between e_i and w_j as

$$d(e_i, w_j) = \begin{cases} 0 & t_s(e_i) \leq t_s(w_j) \leq t_e(e_i) \\ t_e(e_i) - t_s(w_j) & t_s(w_j) > t_e(e_i) \\ t_s(e_i) - t_s(w_j) & t_s(w_j) < t_s(e_i) \end{cases} \quad (4)$$

where $t_s(w_j)$ is the starting timestamp (ms) of word w_j, $t_s(e_i)$ and $t_e(e_i)$ are the starting and ending timestamps (ms) of gaze fixation on entity e_i.

The alignment of word w_j and entity e_i is decided by their temporal distance $d(e_i, w_j)$. Based on the psycholinguistic finding that eye gaze happens before a spoken word, w_j is not allowed to be aligned with e_i when w_j happens earlier than e_i (i.e., $d(e_i, w_j) > 0$). When w_j happens no earlier than e_i (i.e., $d(e_i, w_j) \leq 0$), the closer they are, the more likely they are aligned. Specifically, the temporal alignment probability of w_j and e_i in each co-occurring instance $(\mathbf{w}, \mathbf{e})$ is computed as

$$p_t(a_j = i|j, \mathbf{e}, \mathbf{w}) = \begin{cases} 0 & d(e_i, w_j) > 0 \\ \frac{\exp[\alpha \cdot d(e_i, w_j)]}{\Sigma_i \exp[\alpha \cdot d(e_i, w_j)]} & d(e_i, w_j) \leq 0 \end{cases} \quad (5)$$

where α is a constant for scaling $d(e_i, w_j)$. In our experiments, α is set to 0.005.

An EM algorithm is used to estimate probabilities $p(w|e)$ in Model-2t.

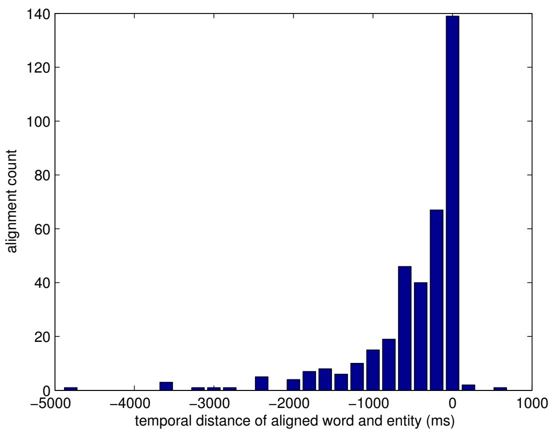


Figure 3: Histogram of truly aligned word and entity pairs over temporal distance (bin width = 200ms)

For the purpose of evaluation, we manually annotated the truly aligned word and entity pairs. Figure 3 shows the histogram of those truly aligned word and entity pairs over the temporal distance of aligned word and entity. We can observe in the figure that 1) almost no eye gaze happens after a spoken word, and 2) the number of word-entity pairs with closer temporal distance is generally larger than the number of those with farther temporal distance. This is consistent with our modeling of the temporal alignment probability of word and entity (Equation (5)).

6 Using Domain Semantic Relatedness for Word Acquisition

Speech-gaze temporal alignment and occurrence statistics sometimes are not sufficient to associate words to an entity correctly. For example, suppose a user says "*there is a lamp on the dresser*" while looking at a lamp object on a table object. Due to their co-occurring with the lamp object, words *dresser* and *lamp* are both likely to be associated with the lamp object in the translation models. As a result, word *dresser* is likely to be incorrectly acquired for the lamp object. For the same reason, the word *lamp* could be acquired incorrectly for the table object. To solve this type of association problem, the semantic knowledge about the domain and words can be helpful. For example, the knowledge that the word *lamp* is more semantically related to the object lamp can help the system avoid associating the word *dresser* to the lamp object. Therefore, we are interested in investigating the use of semantic knowledge in word acquisition.

On one hand, each conversational system has a *domain model*, which is the knowledge representation about its domain such as the types of objects and their properties and relations. On the other hand, there are available resources about domain independent lexical knowledge (e.g., WordNet (Fellbaum, 1998)). The question is whether we can utilize the domain model and external lexical knowledge resource to improve word acquisition. To address this question, we link the domain concepts in the domain model with WordNet concepts, and define semantic relatedness of word and entity to help the system acquire domain semantically compatible words.

In the following sections, we first describe our domain modeling, then define the semantic relatedness of word and entity based on domain modeling and WordNet semantic lexicon, and finally describe different ways of using the semantic relatedness of word and entity to help word acquisition.

6.1 Domain Modeling

We model the 3D room decoration domain as shown in Figure 4. The domain model contains all domain related semantic concepts. These concepts are linked to the WordNet concepts (i.e., synsets in the format of "word#part-of-speech#sense-id"). Each of the entities in the domain has one or more properties (e.g., semantic type, color, size) that are denoted by domain concepts. For example, the entity *dresser_1* has domain concepts *SEM_DRESSER* and *COLOR*. These domain concepts are linked to "dresser#n#4" and "color#n#1" in WordNet.

Note that in the domain model, the domain concepts are not specific to a certain entity, they are general concepts for a certain type of entity. Multiple entities of the same type have the same properties and share the same set of domain concepts.

6.2 Semantic Relatedness of Word and Entity

We compute the semantic relatedness of a word w and an entity e based on the semantic similarity between w and the properties of e. Specifically, semantic relatedness $SR(e, w)$ is defined as

$$SR(e, w) = \max_{i,j} sim(s(c_e^i), s_j(w)) \tag{6}$$

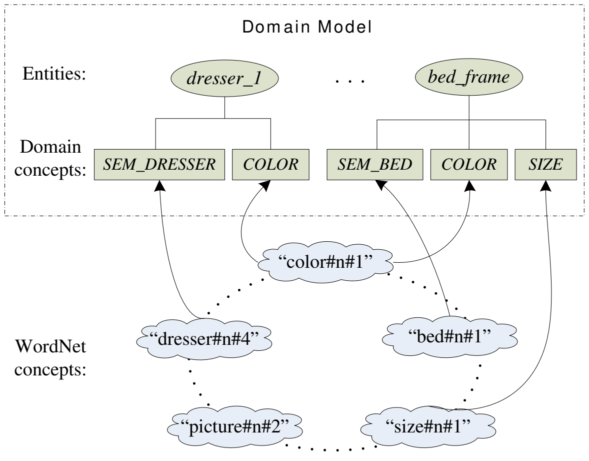


Figure 4: Domain model with domain concepts linked to WordNet synsets

where c_e^i is the i-th property of entity e, $s(c_e^i)$ is the synset of property c_e^i as designed in domain model, $s_j(w)$ is the j-th synset of word w as defined in WordNet, and $sim(\cdot, \cdot)$ is the similarity score of two synsets.

We computed the similarity score of two synsets based on the path length between them. The similarity score is inversely proportional to the number of nodes along the shortest path between the synsets as defined in WordNet. When the two synsets are the same, they have the maximal similarity score of 1. The WordNet-Similarity tool (Pedersen et al., 2004) was used for the synset similarity computation.

6.3 Word Acquisition with Word-Entity Semantic Relatedness

We can use the semantic relatedness of word and entity to help the system acquire semantically compatible words for each entity, and therefore improve word acquisition performance. The semantic relatedness can be applied for word acquisition in two ways: post process learned word-entity association probabilities by rescoring them with semantic relatedness, or directly affect the learning of word-entity associations by constraining the alignment of word and entity in the translation models.

6.3.1 Rescoring with semantic relatedness

In the acquired word list for an entity e_i, each word w_j has an association probability $p(w_j|e_i)$ that is learned from a translation model. We use the semantic relatedness $SR(e_i, w_j)$ to redistribute the probability mass for each w_j. The new association probability is given by:

$$p'(w_j|e_i) = \frac{p(w_j|e_i)SR(e_i, w_j)}{\sum_j p(w_j|e_i)SR(e_i, w_j)} \tag{7}$$

6.3.2 Semantic alignment constraint in translation model

When used to constrain the word-entity alignment in the translation model, semantic relatedness can be used alone or used together with speech-gaze temporal information to decide the alignment probability of word and entity.

- Using only semantic relatedness to constrain word-entity alignments in **Model-2s**, we have

$$p(\mathbf{w}|\mathbf{e}) = \prod_{j=1}^{m} \sum_{i=0}^{l} p_s(a_j = i|j, \mathbf{e}, \mathbf{w}) p(w_j|e_i) \tag{8}$$

  where $p_s(a_j = i|j, \mathbf{e}, \mathbf{w})$ is the alignment probability based on semantic relatedness,

$$p_s(a_j = i|j, \mathbf{e}, \mathbf{w}) = \frac{SR(e_i, w_j)}{\sum_i SR(e_i, w_j)} \tag{9}$$

- Using semantic relatedness and temporal information to constrain word-entity alignments in **Model-2ts**, we have

$$p(\mathbf{w}|\mathbf{e}) = \prod_{j=1}^{m} \sum_{i=0}^{l} p_{ts}(a_j = i|j, \mathbf{e}, \mathbf{w}) p(w_j|e_i) \tag{10}$$

  where $p_{ts}(a_j = i|j, \mathbf{e}, \mathbf{w})$ is the alignment probability that is decided by both temporal relation and semantic relatedness of e_i and w_j,

$$p_{ts}(a_j = i|j, \mathbf{e}, \mathbf{w}) = \frac{p_s(a_j = i|j, \mathbf{e}, \mathbf{w}) p_t(a_j = i|j, \mathbf{e}, \mathbf{w})}{\sum_i p_s(a_j = i|j, \mathbf{e}, \mathbf{w}) p_t(a_j = i|j, \mathbf{e}, \mathbf{w})} \tag{11}$$

  where $p_s(a_j = i|j, \mathbf{e}, \mathbf{w})$ is the semantic alignment probability in Equation (9), and $p_t(a_j = i|j, \mathbf{e}, \mathbf{w})$ is the temporal alignment probability given in Equation (5).

EM algorithms are used to estimate $p(w|e)$ in Model-2s and Model-2ts.

7 Grounding Words to Domain Concepts

As discussed above, based on translation models, we can incorporate temporal and domain semantic information to obtain $p(w|e)$. This probability only provides a means to ground words to entities. In conversational systems, the ultimate goal of word acquisition is to make the system understand the semantic meaning of new words. Word acquisition by grounding words to objects is not always sufficient for identifying their semantic meanings. Suppose the word *green* is grounded to a green chair object, so is the word *chair*. Although the system is aware that *green* is some word describing the green chair, it does not know that word *green* refers to the chair's color while the word *chair* refers to the chair's semantic type. Thus, after learning the word-entity associations $p(w|e)$ by the translation models, we need to further ground words to domain concepts of entity properties.

We further apply WordNet to ground words to domain concepts. For each entity e, based on association probabilities $p(w|e)$, we can choose the n-best words as acquired words for e. Those n-best words have the n highest association probabilities. For each word w acquired for e, the grounded concept c_e^* for w is chosen as the one that has the highest semantic relatedness with w:

$$c_e^* = \arg\max_i \big[\max_j sim(s(c_e^i), s_j(w))\big] \tag{12}$$

where $sim(s(c_e^i), s_j(w))$ is the semantic similarity score defined in Equation (6).

8 Evaluation

We evaluate word acquisition performance of different models on the data collected from our user studies (see Section 3).

8.1 Evaluation Metrics

The following metrics are used to evaluate the words acquired for domain concepts (i.e., entity properties) $\{c_e^i\}$.

- Precision

$$\frac{\sum_e \sum_i \text{\# words correctly acquired for } c_e^i}{\sum_e \sum_i \text{\# words acquired for } c_e^i}$$

- Recall

$$\frac{\sum_e \sum_i \text{\# words correctly acquired for } c_e^i}{\sum_e \sum_i \text{\# ground-truth}^1 \text{ words of } c_e^i}$$

- F-measure

$$\frac{2 \times precision \times recall}{precision + recall}$$

The metrics of precision, recall, and F-measure are based on the n-best words acquired for the entity properties. Therefore, we have different precision, recall, and F-measure when n changes.

The metrics of precision, recall, and F-measure only provide evaluation on the top n candidate words. To measure the acquisition performance on the entire ranked list of candidate words, we define a new metric as follows:

- Mean Reciprocal Rank Rate (MRRR)

$$\text{MRRR} = \frac{\sum_e \frac{\sum_{i=1}^{N_e} \frac{1}{index(w_e^i)}}{\sum_{i=1}^{N_e} \frac{1}{i}}}{\#e}$$

  where N_e is the number of all ground-truth words $\{w_e^i\}$ for entity e, $index(w_e^i)$ is the index of word w_e^i in the ranked list of candidate words for entity e.

Entities may have a different number of ground-truth words. For each entity e, we calculate a Reciprocal Rank Rate (RRR), which measures how close the ranks of the ground-truth words in the candidate word list is to the best scenario where the top N_e words are the ground-truth words for e. RRR is in the range of $(0, 1]$. The higher the RRR, the better is the word acquisition performance. The average of RRRs across all entities gives the Mean Reciprocal Rank Rate (MRRR).

Note that MRRR is directly based on the learned word-entity associations $p(w|e)$, it is in fact a measure of grounding words to entities.

[1]The ground-truth words were compiled and agreed upon by two human judges.

8.2 Evaluation Results

To compare the effects of different speech-gaze alignments on word acquisition, we evaluate the following models:

- Model-1 – base model I without word-entity alignment (Equation (1)).
- Model-2 – base model II with positional alignment (Equation (2)).
- Model-2t – enhanced model with temporal alignment (Equation (3)).
- Model-2s – enhanced model with semantic alignment (Equation (8)).
- Model-2ts – enhanced model with both temporal and semantic alignment (Equation (10)).

To compare the different ways of incorporating semantic relatedness in word acquisition as discussed in Section 6.3.1, we also evaluate the following models:

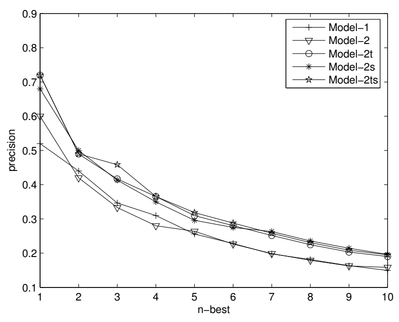
- Model-1-r – Model-1 with semantic relatedness rescoring of word-entity association.
- Model-2t-r – Model-2t with semantic relatedness rescoring of word-entity association.

Figure 5 shows the results of models with different speech-gaze alignments. Figure 6 shows the results of models with semantic relatedness rescoring. In Figure 5 & 6, *n-best* means the top *n* word candidates are chosen as acquired words for each entity. The Mean Reciprocal Rank Rates of all models are compared in Figure 7.

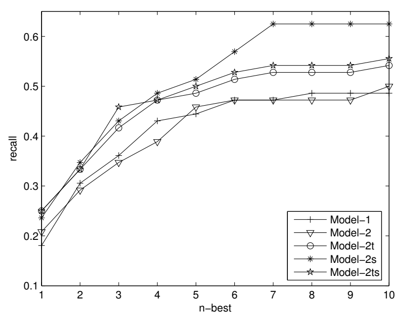
8.2.1 Results of using different speech-gaze alignments

As shown in Figure 5, Model-2 does not show a consistent improvement compared to Model-1 when a different number of n-best words are chosen as acquired words. This result shows that it is not very helpful to consider the index-based positional alignment of word and entity for word acquisition.

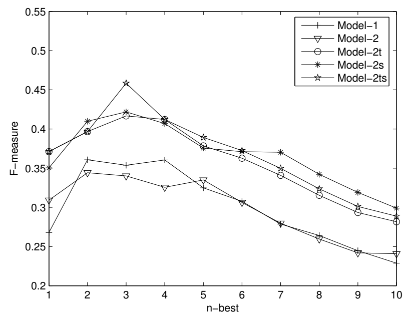
Figure 5 also shows that models considering temporal or/and semantic information (Model-2t, Model-2s, Model-2ts) consistently perform better than the models considering neither temporal nor



(a) precision

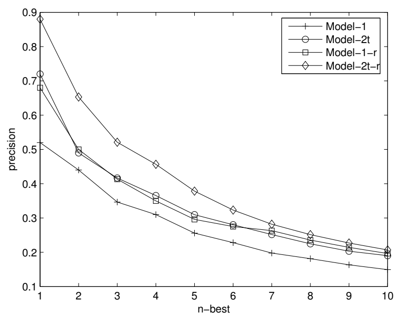


(b) recall

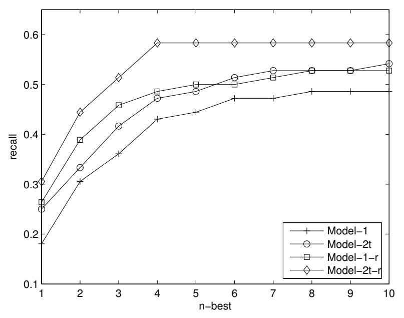


(c) F-measure

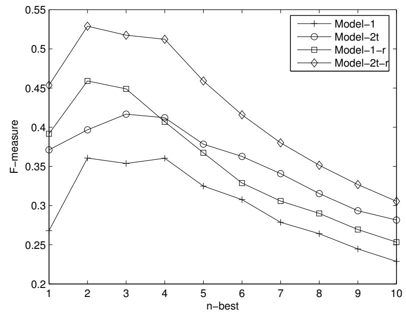
Figure 5: Performance of word acquisition when different types of speech-gaze alignment are applied



(a) precision



(b) recall



(c) F-measure

Figure 6: Performance of word acquisition when semantic relatedness rescoring of word-entity association is applied

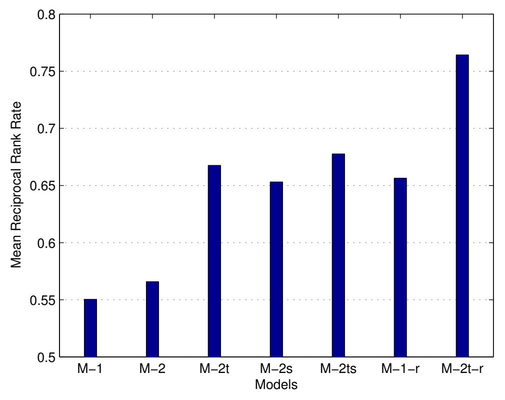


Figure 7: MRRRs achieved by different models

semantic information (Model-1, Model-2). Among Model-2t, Model-2s, and Model-2ts, it is found that they do not make consistent differences.

As shown in Figure 7, the MRRRs of different models are consistent with their performances on F-measure. A t-test has shown that the difference between the MRRRs of Model-1 and Model-2 is not statistically significant. Compared to Model-1, t-tests have confirmed that MRRR is significantly improved by Model-2t ($t = 2.27, p < 0.02$), Model-2s ($t = 3.40, p < 0.01$), and Model-2ts($t = 2.60, p < 0.01$). T-tests have shown no significant differences among Model-2t, Model-2s, and Model-2ts.

8.2.2 Results of applying semantic relatedness rescoring

Figure 6 shows that semantic relatedness rescoring improves word acquisition. After semantic relatedness rescoring of the word-entity associations learned by Model-1, Model-1-r improves the F-measure consistently when a different number of n-best words are chosen as acquired words. Compared to Model-2t, Model-2t-r also improves the F-measure consistently.

Comparing the two ways of using semantic relatedness for word acquisition, it is found that rescoring word-entity association with semantic relatedness works better. When semantic relatedness is used together with temporal information to constrain word-entity alignments in Model-2ts, word acqui-

Model	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5
M-1	**table**(0.173)	**dresser**(0.067)	area(0.058)	picture(0.053)	dressing(0.041)
M-2t	**table**(0.146)	**dresser**(0.125)	dressing(0.061)	**vanity**(0.051)	fact(0.050)
M-2t-r	**table**(0.312)	**dresser**(0.241)	**vanity**(0.149)	**desk**(0.047)	area(0.026)

Table 1: N-best candidate words acquired for the entity *dresser_1* by different models

sition performance is not improved compared to Model-2t. However, using semantic relatedness to rescore word-entity association learned by Model-2t, Model-2t-r further improves word acquisition.

As shown in Figure 7, the MRRRs of Model-1-r and Model-2t-r are consistent with their performances on F-measure. Compared to Model-2t, Model-2t-r improves MRRR. A t-test has confirmed that this is a significant improvement ($t = 1.97, p < 0.03$). Compared to Model-1, Model-1-r significantly improves MRRR ($t = 2.33, p < 0.02$). There is no significant difference between Model-1-r and Model-2t/Model-2s/Model-2ts.

In Figures 5&6, we also notice that the recall of the acquired words is still comparably low even when 10 best word candidates are chosen for each entity. This is mainly due to the scarcity of those words that are not acquired in the data. Many of the words that are not acquired appear less than 3 times in the data, which makes them unlikely to be associated with any entity by the translation models. When more data is available, we expect to see higher recall.

8.3 An Example

Table 1 shows the 5-best words acquired by different models for the entity *dresser_1* in the 3d room scene (see Figure 1). In the table, each word is followed by its word-entity association probability $p(w|e)$. The correctly acquired words are shown in bold font.

As shown in the example, the baseline Model-1 learned 2 correct words in the 5-best list. Considering speech-gaze temporal information, Model-2t learned one more correct word *vanity* in the 5-best list. With semantic relatedness rescoring, Model-2t-r further acquired word *desk* in the 5-best list because of the high semantic relatedness of word *desk* and the type of entity *dresser_1*. Although neither Model-1 nor Model-2t successfully acquired the word *desk* in the 5-best list, the rank (=7) of the word *desk* in Model-2t's n-best list is much higher than the rank (=21) in Model-1's n-best list.

9 Conclusion

Motivated by the psycholinguistic findings, we investigate the use of eye gaze for automatic word acquisition in multimodal conversational systems. Particularly, we investigate the use of speech-gaze temporal information and word-entity semantic relatedness to facilitate word acquisition. Our experiments show that word acquisition is significantly improved when temporal information is considered, which is consistent with the previous psycholinguistic findings about speech and eye gaze. Moreover, using temporal information together with semantic relatedness rescoring further improves word acquisition.

Eye tracking systems are no longer bulky systems that prevent natural human machine communication. Display mounted gaze tracking systems (e.g., Tobii) are completely non-intrusive, can tolerate head motion, and provide high tracking quality. Integrating eye tracking with conversational interfaces is no longer beyond reach. Recent works have shown that eye gaze can facilitate spoken language processing in conversational systems (Qu and Chai, 2007; Prasov and Chai, 2008). Incorporating eye gaze with automatic word acquisition provides another potential approach to improve the robustness of human machine conversation.

Acknowledgments

This work was supported by IIS-0347548 and IIS-0535112 from the National Science Foundation. The authors would like to thank Zahar Prasov for his contribution on data collection. The authors would also like to thank anonymous reviewers for their valuable comments and suggestions.

References

Kobus Barnard, Pinar Duygulu, Nando de Freitas, David Forsyth, David Blei, and Michael I. Jordan. 2003.

Matching words and pictures. *Journal of Machine Learning Research*, 3:1107–1135.

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

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Zenzi M. Griffin and Kathryn Bock. 2000. What the eyes say about speaking. *Psychological Science*, 11:274–279.

John M. Henderson and Fernanda Ferreira, editors. 2004. *The interface of language, vision, and action: Eye movements and the visual world*. New York: Taylor & Francis.

Marcel A. Just and Patricia A. Carpenter. 1976. Eye fixations and cognitive processes. *Cognitive Psychology*, 8:441–480.

Yi Liu, Joyce Y. Chai, and Rong Jin. 2007. Automated vocabulary acquisition and interpretation in multimodal conversational systems. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL)*.

E. Matin. 1974. Saccadic suppression: a review and an analysis. *Psychological Bulletin*, 81:899–917.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet::similarity - measuring the relatedness of concepts. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*.

Zahar Prasov and Joyce Y. Chai. 2008. What's in a gaze? the role of eye-gaze in reference resolution in multimodal conversational interfaces. In *Proceedings of ACM 12th International Conference on Intelligent User interfaces (IUI)*.

Shaolin Qu and Joyce Y. Chai. 2007. An exploration of eye gaze in spoken language processing for multimodal conversational interfaces. In *Proceedings of the Conference of the North America Chapter of the Association of Computational Linguistics (NAACL)*.

Deb K. Roy and Alex P. Pentland. 2002. Learning words from sights and sounds, a computational model. *Cognitive Science*, 26(1):113–146.

Michael K. Tanenhaus, Michael J. Spivey-Knowiton, Kathleen M. Eberhard, and Julie C. Sedivy. 1995. Integration of visual and linguistic information in spoken language comprehension. *Science*, 268:1632–1634.

Chen Yu and Dana H. Ballard. 2004. A multimodal learning interface for grounding spoken language in sensory perceptions. *ACM Transactions on Applied Perceptions*, 1(1):57–80.

Cheap and Fast — But is it Good?
Evaluating Non-Expert Annotations for Natural Language Tasks

Rion Snow[†] **Brendan O'Connor**[‡] **Daniel Jurafsky**[§] **Andrew Y. Ng**[†]

[†]Computer Science Dept.
Stanford University
Stanford, CA 94305
{rion,ang}@cs.stanford.edu

[‡]Dolores Labs, Inc.
832 Capp St.
San Francisco, CA 94110
brendano@doloreslabs.com

[§]Linguistics Dept.
Stanford University
Stanford, CA 94305
jurafsky@stanford.edu

Abstract

Human linguistic annotation is crucial for many natural language processing tasks but can be expensive and time-consuming. We explore the use of Amazon's Mechanical Turk system, a significantly cheaper and faster method for collecting annotations from a broad base of paid non-expert contributors over the Web. We investigate five tasks: affect recognition, word similarity, recognizing textual entailment, event temporal ordering, and word sense disambiguation. For all five, we show high agreement between Mechanical Turk non-expert annotations and existing gold standard labels provided by expert labelers. For the task of affect recognition, we also show that using non-expert labels for training machine learning algorithms can be as effective as using gold standard annotations from experts. We propose a technique for bias correction that significantly improves annotation quality on two tasks. We conclude that many large labeling tasks can be effectively designed and carried out in this method at a fraction of the usual expense.

1 Introduction

Large scale annotation projects such as TreeBank (Marcus et al., 1993), PropBank (Palmer et al., 2005), TimeBank (Pustejovsky et al., 2003), FrameNet (Baker et al., 1998), SemCor (Miller et al., 1993), and others play an important role in natural language processing research, encouraging the development of novel ideas, tasks, and algorithms. The construction of these datasets, however, is extremely expensive in both annotator-hours and financial cost. Since the performance of many natural language processing tasks is limited by the amount and quality of data available to them (Banko and Brill, 2001), one promising alternative for some tasks is the collection of non-expert annotations.

In this work we explore the use of Amazon Mechanical Turk[1] (AMT) to determine whether non-expert labelers can provide reliable natural language annotations. We chose five natural language understanding tasks that we felt would be sufficiently natural and learnable for non-experts, and for which we had gold standard labels from expert labelers, as well as (in some cases) expert labeler agreement information. The tasks are: affect recognition, word similarity, recognizing textual entailment, event temporal ordering, and word sense disambiguation. For each task, we used AMT to annotate data and measured the quality of the annotations by comparing them with the gold standard (expert) labels on the same data. Further, we compare machine learning classifiers trained on expert annotations vs. non-expert annotations.

In the next sections of the paper we introduce the five tasks and the evaluation metrics, and offer methodological insights, including a technique for bias correction that improves annotation quality.[2]

[1] http://mturk.com

[2] Please see http://blog.doloreslabs.com/?p=109 for a condensed version of this paper, follow-ups, and ongoing public discussion. We encourage comments to be directed here in addition to email when appropriate. Dolores Labs Blog, "AMT is fast, cheap, and good for machine learning data," Brendan O'Connor, Sept. 9, 2008. More related work at http://blog.doloreslabs.com/topics/wisdom/.

2 Related Work

The idea of collecting annotations from volunteer contributors has been used for a variety of tasks. Luis von Ahn pioneered the collection of data via online annotation tasks in the form of games, including the ESPGame for labeling images (von Ahn and Dabbish, 2004) and Verbosity for annotating word relations (von Ahn et al., 2006). The Open Mind Initiative (Stork, 1999) has taken a similar approach, attempting to make such tasks as annotating word sense (Chklovski and Mihalcea, 2002) and commonsense word relations (Singh, 2002) sufficiently "easy and fun" to entice users into freely labeling data.

There have been an increasing number of experiments using Mechanical Turk for annotation. In (Su et al., 2007) workers provided annotations for the tasks of hotel name entity resolution and attribute extraction of age, product brand, and product model, and were found to have high accuracy compared to gold-standard labels. Kittur et al. (2008) compared AMT evaluations of Wikipedia article quality against experts, finding validation tests were important to ensure good results. Zaenen (Submitted) studied the agreement of annotators on the problem of recognizing textual entailment (a similar task and dataset is explained in more detail in Section 4).

At least several studies have already used AMT without external gold standard comparisons. In (Nakov, 2008) workers generated paraphrases of 250 noun-noun compounds which were then used as the gold standard dataset for evaluating an automatic method of noun compound paraphrasing. Kaisser and Lowe (2008) use AMT to help build a dataset for question answering, annotating the answers to 8107 questions with the *sentence* containing the answer. Kaisser et al. (2008) examines the task of customizing the summary length of QA output; non-experts from AMT chose a summary length that suited their information needs for varying query types. Dakka and Ipeirotis (2008) evaluate a document facet generation system against AMT-supplied facets, and also use workers for user studies of the system. Sorokin and Forsyth (2008) collect data for machine vision tasks and report speed and costs similar to our findings; their summaries of worker behavior also corroborate with what we have found.

In general, volunteer-supplied or AMT-supplied data is more plentiful but noisier than expert data. It is powerful because independent annotations can be aggregated to achieve high reliability. Sheng et al. (2008) explore several methods for using many noisy labels to create labeled data, how to choose which examples should get more labels, and how to include labels' uncertainty information when training classifiers. Since we focus on empirically validating AMT as a data source, we tend to stick to simple aggregation methods.

3 Task Design

In this section we describe Amazon Mechanical Turk and the general design of our experiments.

3.1 Amazon Mechanical Turk

We employ the Amazon Mechanical Turk system in order to elicit annotations from non-expert labelers. AMT is an online labor market where workers are paid small amounts of money to complete small tasks. The design of the system is as follows: one is required to have an Amazon account to either submit tasks for annotations or to annotate submitted tasks. These Amazon accounts are anonymous, but are referenced by a unique Amazon ID. A *Requester* can create a *group* of *Human Intelligence Tasks* (or *HIT*s), each of which is a form composed of an arbitrary number of questions. The user requesting annotations for the group of HITs can specify the number of unique annotations per HIT they are willing to pay for, as well as the reward payment for each individual HIT. While this does not guarantee that unique people will annotate the task (since a single person could conceivably annotate tasks using multiple accounts, in violation of the user agreement), this does guarantee that annotations will be collected from unique accounts. AMT also allows a requester to restrict which workers are allowed to annotate a task by requiring that all workers have a particular set of qualifications, such as sufficient accuracy on a small test set or a minimum percentage of previously accepted submissions. Annotators (variously referred to as *Workers* or *Turkers*) may then annotate the tasks of their choosing. Finally, after each HIT has been annotated, the Requester has the option of approving the work and optionally giving a bonus to individual workers. There is a two-way communication channel between the task designer and the workers mediated by Amazon, and Amazon handles all financial transactions.

3.2 Task Design

In general we follow a few simple design principles: we attempt to keep our task descriptions as succinct as possible, and we attempt to give demonstrative examples for each class wherever possible. We have published the full experimental design and the data we have collected for each task online[3]. We have restricted our study to tasks where we require only a multiple-choice response or numeric input within a fixed range. For every task we collect ten independent annotations for each unique item; this redundancy allows us to perform an in-depth study of how data quality improves with the number of independent annotations.

4 Annotation Tasks

We analyze the quality of non-expert annotations on five tasks: affect recognition, word similarity, recognizing textual entailment, temporal event recognition, and word sense disambiguation. In this section we define each annotation task and the parameters of the annotations we request using AMT. Additionally we give an initial analysis of the task results, and summarize the cost of the experiments.

4.1 Affective Text Analysis

This experiment is based on the affective text annotation task proposed in Strapparava and Mihalcea (2007), wherein each annotator is presented with a list of short headlines, and is asked to give numeric judgments in the interval [0,100] rating the headline for six emotions: anger, disgust, fear, joy, sadness, and surprise, and a single numeric rating in the interval [-100,100] to denote the overall positive or negative *valence* of the emotional content of the headline, as in this sample headline-annotation pair:

Outcry at N Korea 'nuclear test'

$(Anger, 30), (Disgust,30), (Fear,30), (Joy,0),$
$(Sadness,20), (Surprise,40), (Valence,-50).$

[3]All tasks and collected data are available at `http://ai.stanford.edu/~rion/annotations/`.

For our experiment we select a 100-headline sample from the original SemEval test set, and collect 10 affect annotations for each of the seven label types, for a total of 7000 affect labels.

We then performed two comparisons to evaluate the quality of the AMT annotations. First, we asked how well the non-experts agreed with the experts. We did this by comparing the interannotator agreement (ITA) of individual expert annotations to that of single non-expert and averaged non-expert annotations. In the original experiment ITA is measured by calculating the Pearson correlation of one annotator's labels with the average of the labels of the other five annotators. For each expert labeler, we computed this ITA score of the expert against the other five; we then average these ITA scores across all expert annotators to compute the average expert ITA (reported in Table 1 as "E vs. E". We then do the same for individual non-expert annotations, averaging Pearson correlation across all sets of the five expert labelers ("NE vs. E"). We then calculate the ITA for each expert vs. the averaged labels from all other experts and non-experts (marked as "E vs. All") and for each non-expert vs. the pool of other non-experts and all experts ("NE vs. All"). We compute these ITA scores for each emotion task separately, averaging the six emotion tasks as "Avg. Emo" and the average of all tasks as "Avg. All".

Emotion	E vs. E	E vs. All	NE vs. E	NE vs. All
Anger	0.459	0.503	0.444	0.573
Disgust	0.583	0.594	0.537	0.647
Fear	0.711	0.683	0.418	0.498
Joy	0.596	0.585	0.340	0.421
Sadness	0.645	0.650	0.563	0.651
Surprise	0.464	0.463	0.201	0.225
Valence	0.759	0.767	0.530	0.554
Avg. Emo	0.576	0.603	0.417	0.503
Avg. All	0.580	0.607	0.433	0.510

Table 1: Average expert and non-expert ITA on test-set

The results in Table 1 conform to the expectation that experts are better labelers: experts agree with experts more than non-experts agree with experts, although the ITAs are in many cases quite close. But we also found that adding non-experts to the gold standard ("E vs. All") improves agreement, suggesting that non-expert annotations are good enough to increase the overall quality of the gold labels. Our first comparison showed that individual experts were better than individual non-experts. In our next comparison we ask how many averaged non-experts it would take to rival the performance of a single expert. We did this by averaging the labels of each possible subset of n non-expert annotations, for value of n in $\{1, 2, \ldots, 10\}$. We then treat this average as though it is the output of a single 'meta-labeler', and compute the ITA with respect to each subset of five of the six expert annotators. We then average the results of these studies across each subset size; the results of this experiment are given in Table 2 and in Figure 1. In addition to the single meta-labeler, we ask: what is the minimum number of non-expert annotations k from which we can create a meta-labeler that has equal or better ITA than an expert annotator? In Table 2 we give the minimum k for each emotion, and the averaged ITA for that meta-labeler consisting of k non-experts (marked "k-NE"). In Figure 1 we plot the expert ITA correlation as the horizontal dashed line.

Emotion	1-Expert	10-NE	k	k-NE
Anger	0.459	0.675	2	0.536
Disgust	0.583	0.746	2	0.627
Fear	0.711	0.689	–	–
Joy	0.596	0.632	7	0.600
Sadness	0.645	0.776	2	0.656
Surprise	0.464	0.496	9	0.481
Valence	0.759	0.844	5	0.803
Avg. Emo.	0.576	0.669	4	0.589
Avg. All	0.603	0.694	4	0.613

Table 2: Average expert and averaged correlation over 10 non-experts on test-set. k is the minimum number of non-experts needed to beat an average expert.

These results show that for all tasks except "Fear" we are able to achieve expert-level ITA with the held-out set of experts within 9 labelers, and frequently within only 2 labelers. Pooling judgments across all 7 tasks we find that on average it requires only 4 non-expert annotations per example to achieve the equivalent ITA as a single expert annotator. Given that we paid US\$2.00 in order to collect the 7000 non-expert annotations, we may interpret our rate of 3500 non-expert labels per USD as at least 875 expert-equivalent labels per USD.

4.2 Word Similarity

This task replicates the word similarity task used in (Miller and Charles, 1991), following a previous

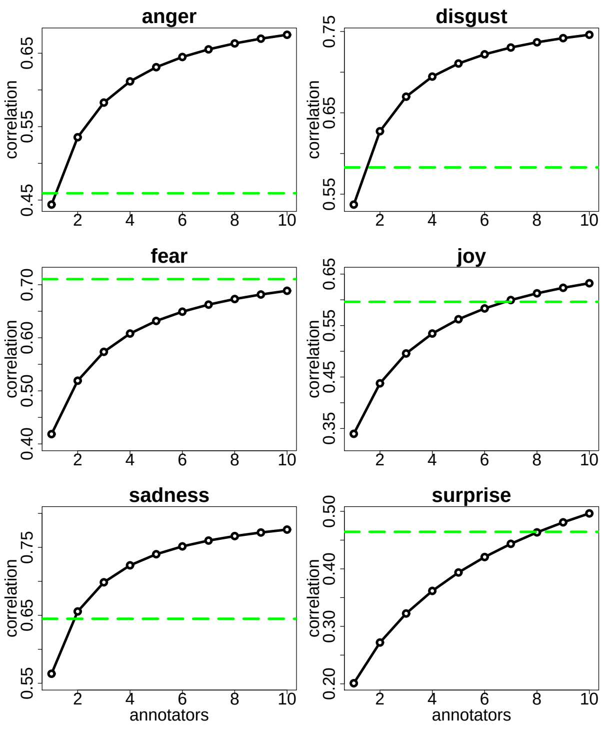


Figure 1: Non-expert correlation for affect recognition

task initially proposed by (Rubenstein and Goodenough, 1965). Specifically, we ask for numeric judgments of word similarity for 30 word pairs on a scale of [0,10], allowing fractional responses[4]. These word pairs range from highly similar (e.g., {*boy, lad*}), to unrelated (e.g., {*noon, string*}). Numerous expert and non-expert studies have shown that this task typically yields very high interannotator agreement as measured by Pearson correlation; (Miller and Charles, 1991) found a 0.97 correlation of the annotations of 38 subjects with the annotations given by 51 subjects in (Rubenstein and Goodenough, 1965), and a following study (Resnik, 1999) with 10 subjects found a 0.958 correlation with (Miller and Charles, 1991).

In our experiment we ask for 10 annotations each of the full 30 word pairs, at an offered price of \$0.02 for each set of 30 annotations (or, equivalently, at the rate of 1500 annotations per USD). The most surprising aspect of this study was the speed with which it was completed; the task of 300 annotations was completed by 10 annotators in less than 11 min-

[4](Miller and Charles, 1991) and others originally used a numerical score of [0,4].

utes from the time of submission of our task to AMT, at the rate of 1724 annotations / hour.

As in the previous task we evaluate our non-expert annotations by averaging the numeric responses from each possible subset of n annotators and computing the interannotator agreement with respect to the gold scores reported in (Miller and Charles, 1991). Our results are displayed in Figure 2, with Resnik's 0.958 correlation plotted as the horizontal line; we find that at 10 annotators we achieve a correlation of 0.952, well within the range of other studies of expert and non-expert annotations.

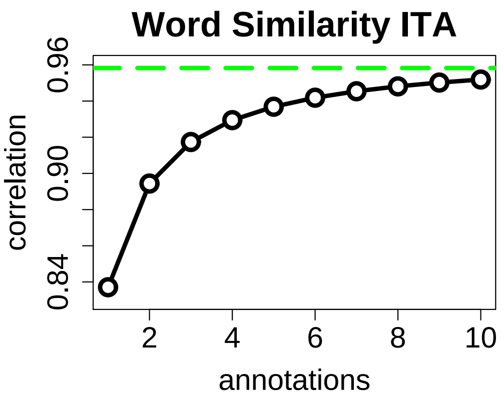


Figure 2: ITA for word similarity experiment

4.3 Recognizing Textual Entailment

This task replicates the recognizing textual entailment task originally proposed in the PASCAL Recognizing Textual Entailment task (Dagan et al., 2006); here for each question the annotator is presented with two sentences and given a binary choice of whether the second *hypothesis* sentence can be inferred from the first. For example, the hypothesis sentence "*Oil prices drop*" would constitute a *true entailment* from the text "*Crude Oil Prices Slump*", but a *false entailment* from "*The government announced last week that it plans to raise oil prices*".

We gather 10 annotations each for all 800 sentence pairs in the PASCAL RTE-1 dataset. For this dataset expert interannotator agreement studies have been reported as achieving 91% and 96% agreement over various subsections of the corpus. When considering multiple non-expert annotations for a sentence pair we use simple majority voting, breaking ties randomly and averaging performance over all possible ways to break ties. We collect 10 annotations for each of 100 RTE sentence pairs; as displayed in Figure 3, we achieve a maximum accuracy of 89.7%, averaging over the annotations of 10 workers[5].

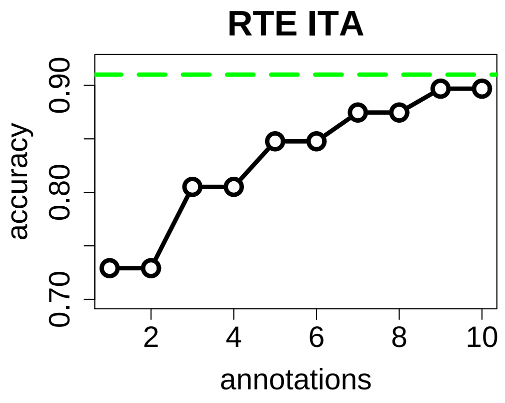


Figure 3: Inter-annotator agreement for RTE experiment

4.4 Event Annotation

This task is inspired by the TimeBank corpus (Pustejovsky et al., 2003), which includes among its annotations a label for event-pairs that represents the temporal relation between them, from a set of fourteen relations (*before*, *after*, *during*, *includes*, etc.). We implement *temporal ordering* as a simplified version of the TimeBank event temporal annotation task: rather than annotating all fourteen event types, we restrict our consideration to the two simplest labels: "strictly before" and "strictly after". Furthermore, rather than marking both nouns and verbs in the text as possible events, we only consider possible verb events. We extract the 462 verb event pairs labeled as "strictly before" or "strictly after" in the TimeBank corpus, and we present these pairs to annotators with a forced binary choice on whether the event described by the first verb occurs *before* or *after* the second. For example, in a dialogue about a plane explosion, we have the utterance: "*It just blew up in the air, and then we saw two fireballs go down to the,*

[5]It might seem pointless to consider an even number of annotations in this circumstance, since the majority voting mechanism and tie-breaking yields identical performance for $2n + 1$ and $2n + 2$ annotators; however, in Section 5 we will consider methods that can make use of the even annotations.

to the water, and there was a big small, ah, smoke, from ah, coming up from that". Here for each annotation we highlight the specific verb pair of interest (e.g., *go*/*coming*, or *blew*/*saw*) and ask which event occurs first (here, *go* and *blew*, respectively).

The results of this task are presented in Figure 4. We achieve high agreement for this task, at a rate of 0.94 with simple voting over 10 annotators (4620 total annotations). While an expert ITA of 0.77 was reported for the more general task involving all fourteen labels on both noun and verb events, no expert ITA numbers have been reported for this simplified temporal ordering task.

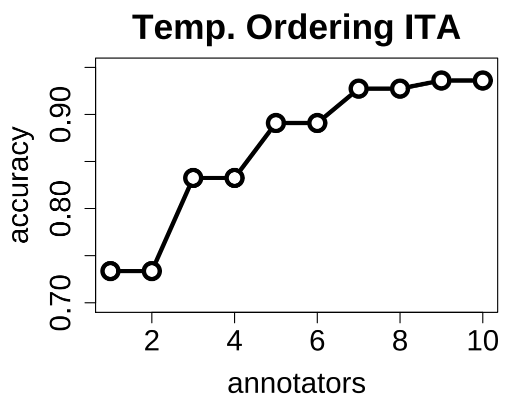


Figure 4: ITA for temporal ordering experiment

4.5 Word Sense Disambiguation

In this task we consider a simple problem on which machine learning algorithms have been shown to produce extremely good results; here we annotate part of the SemEval Word Sense Disambiguation Lexical Sample task (Pradhan et al., 2007); specifically, we present the labeler with a paragraph of text containing the word "president" (e.g., a paragraph containing "*Robert E. Lyons III...was appointed president and chief operating officer...*") and ask the labeler which one of the following three sense labels is most appropriate:

1) *executive officer of a firm, corporation, or university*
2) *head of a country (other than the U.S.)*
3) *head of the U.S., President of the United States*

We collect 10 annotations for each of 177 examples of the noun "president" for the three senses given in SemEval. As shown in Figure 5, performing simple majority voting (with random tie-breaking) over annotators results in a rapid accuracy plateau at a very high rate of 0.994 accuracy. In fact, further analysis reveals that there was only a single disagreement between the averaged non-expert vote and the gold standard; on inspection it was observed that the annotators voted strongly against the original gold label (9-to-1 against), and that it was in fact found to be an error in the original gold standard annotation.[6] After correcting this error, the non-expert accuracy rate is 100% on the 177 examples in this task. This is a specific example where non-expert annotations can be used to correct expert annotations.

Since expert ITA was not reported per word on this dataset, we compare instead to the performance of the best automatic system performance for disambiguating "president" in SemEval Task 17 (Cai et al., 2007), with an accuracy of 0.98.

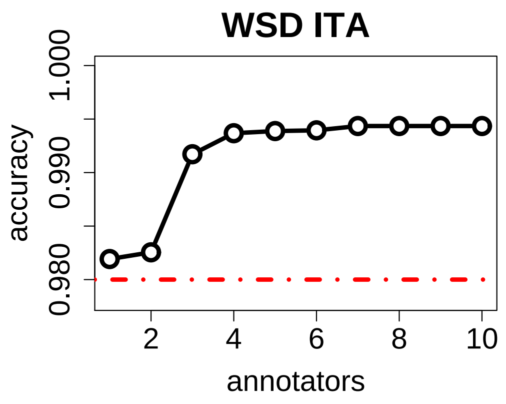


Figure 5: Inter-annotator agreement for WSD experiment

4.6 Summary

Task	Labels	Cost (USD)	Time (hrs)	Labels per USD	Labels per hr
Affect	7000	$2.00	5.93	3500	1180.4
WSim	300	$0.20	0.174	1500	1724.1
RTE	8000	$8.00	89.3	1000	89.59
Event	4620	$13.86	39.9	333.3	115.85
WSD	1770	$1.76	8.59	1005.7	206.1
Total	21690	25.82	143.9	840.0	150.7

Table 3: Summary of costs for non-expert labels

[6]The example sentence began "*The Egyptian president said he would visit Libya today...*" and was mistakenly marked as the "head of a company" sense in the gold annotation (example id 24:0@24@wsj/23/wsj_2381@wsj@en@on).

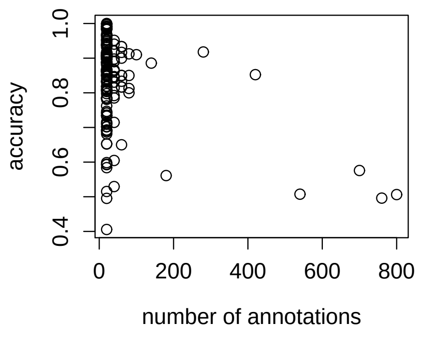


Figure 6: Worker accuracies on the RTE task. Each point is one worker. Vertical jitter has been added to points on the left to show the large number of workers who did the minimum amount of work (20 examples).

In Table 3 we give a summary of the costs associated with obtaining the non-expert annotations for each of our 5 tasks. Here *Time* is given as the total amount of time in hours elapsed from submitting the group of HITs to AMT until the last assignment is submitted by the last worker.

5 Bias correction for non-expert annotators

The reliability of individual workers varies. Some are very accurate, while others are more careless and make mistakes; and a small few give very noisy responses. Furthermore, for most AMT data collection experiments, a relatively small number of workers do a large portion of the task, since workers may do as much or as little as they please. Figure 6 shows accuracy rates for individual workers on one task. Both the overall variability, as well as the prospect of identifying high-volume but low-quality workers, suggest that controlling for individual worker quality could yield higher quality overall judgments.

In general, there are at least three ways to enhance quality in the face of worker error. More workers can be used, as described in previous sections. Another method is to use Amazon's compensation mechanisms to give monetary bonuses to highly-performing workers and deny payments to unreliable ones; this is useful, but beyond the scope of this paper. In this section we explore a third alternative, to model the reliability and biases of individual workers and correct for them.

A wide number of methods have been explored to correct for the bias of annotators. Dawid and Skene (1979) are the first to consider the case of having multiple annotators per example but unknown true labels. They introduce an EM algorithm to simultaneously estimate annotator biases and latent label classes. Wiebe et al. (1999) analyze linguistic annotator agreement statistics to find bias, and use a similar model to correct labels. A large literature in biostatistics addresses this same problem for medical diagnosis. Albert and Dodd (2004) review several related models, but argue they have various shortcomings and emphasize instead the importance of having a gold standard.

Here we take an approach based on gold standard labels, using a small amount of expert-labeled training data in order to correct for the individual biases of different non-expert annotators. The idea is to recalibrate worker's responses to more closely match expert behavior. We focus on categorical examples, though a similar method can be used with numeric data.

5.1 Bias correction in categorical data

Following Dawid and Skene, we model labels and workers with a multinomial model similar to Naive Bayes. Every example i has a true label x_i. For simplicity, assume two labels $\{Y, N\}$. Several different workers give labels $y_{i1}, y_{i2}, \dots y_{iW}$. A worker's conditional probability of response is modeled as multinomial, and we model each worker's judgment as conditionally independent of other workers given the true label x_i, i.e.:

$$P(y_{i1}, \dots, y_{iW}, x_i) = \left(\prod_w P(y_{iw}|x_i) \right) p(x_i)$$

To infer the posterior probability of the true label for a new example, worker judgments are integrated via Bayes rule, yielding the posterior log-odds:

$$\begin{aligned} & \log \frac{P(x_i = Y|y_{i1} \dots y_{iW})}{P(x_i = N|y_{i1} \dots y_{iW})} \\ = \quad & \sum_w \log \frac{P(y_{iw}|x_i = Y)}{P(y_{iw}|x_i = N)} + \log \frac{P(x_i = Y)}{P(x_i = N)} \end{aligned}$$

The worker response likelihoods $P(y_w|x = Y)$ and $P(y_w|x = N)$ can be directly estimated from frequencies of worker performance on gold standard examples. (If we used maximum likelihood estimation with no Laplace smoothing, then each $y_w|x$ is just the worker's empirical confusion matrix.) For MAP label estimation, the above equation describes a weighted voting rule: each worker's vote is weighted by their log likelihood ratio for their given response. Intuitively, workers who are more than 50% accurate have positive votes; workers whose judgments are pure noise have zero votes; and anticorrelated workers have negative votes. (A simpler form of the model only considers accuracy rates, thus weighting worker votes by $\log \frac{\text{acc}_w}{1-\text{acc}_w}$. But we use the full unconstrained multinomial model here.)

5.1.1 Example tasks: RTE-1 and event annotation

We used this model to improve accuracy on the RTE-1 and event annotation tasks. (The other categorical task, word sense disambiguation, could not be improved because it already had maximum accuracy.) First we took a sample of annotations giving k responses per example. Within this sample, we trained and tested via 20-fold cross-validation across examples. Worker models were fit using Laplace smoothing of 1 pseudocount; label priors were uniform, which was reasonably similar to the empirical distribution for both tasks.

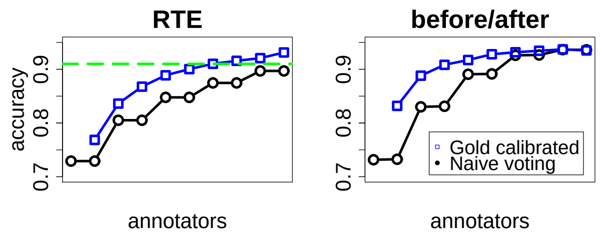


Figure 7: Gold-calibrated labels versus raw labels

Figure 7 shows improved accuracy at different numbers of annotators. The lowest line is for the naive 50% majority voting rule. (This is equivalent to the model under uniform priors and equal accuracies across workers and labels.) Each point is the data set's accuracy against the gold labels, averaged across resamplings each of which obtains k annotations per example. RTE has an average +4.0% accuracy increase, averaged across 2 through 10 annotators. We find a +3.4% gain on event annotation. Finally, we experimented with a similar calibration method for numeric data, using a Gaussian noise model for each worker: $y_w|x \sim N(x + \mu_w, \sigma_w)$. On the affect task, this yielded a small but consistent increases in Pearson correlation at all numbers of annotators, averaging a +0.6% gain.

6 Training a system with non-expert annotations

In this section we train a supervised affect recognition system with expert vs. non-expert annotations.

6.1 Experimental Design

For the purpose of this experiment we create a simple bag-of-words unigram model for predicting affect and valence, similar to the SWAT system (Katz et al., 2007), one of the top-performing systems on the SemEval Affective Text task.[7] For each token t in our training set, we assign t a weight for each emotion e equal to the average emotion score observed in each headline H that t participates in. i.e., if $\mathbf{H}_t$ is the set of headlines containing the token t, then:

$$Score(e,t) = \frac{\sum_{H \in \mathbf{H}_t} Score(e,H)}{|\mathbf{H}_t|}$$

With these weights of the individual tokens we may then compute the score for an emotion e of a new headline H as the average score over the set of tokens $t \in H$ that we've observed in the training set (ignoring those tokens not in the training set), i.e.:

$$Score(e,H) = \sum_{t \in H} \frac{Score(e,t)}{|H|}$$

Where $|H|$ is simply the number of tokens in headline H, ignoring tokens not observed in the training set.

[7] Unlike the SWAT system we perform no lemmatization, synonym expansion, or any other preprocessing of the tokens; we simply use whitespace-separated tokens within each headline.

6.2 Experiments

We use 100 headlines as a training set (examples 500-599 from the test set of SemEval Task 14), and we use the remaining 900 headlines as our test set. Since we are fortunate to have the six separate expert annotations in this task, we can perform an extended systematic comparison of the performance of the classifier trained with expert vs. non-expert data.

Emotion	1-Expert	10-NE	k	k-NE
Anger	0.084	0.233	1	0.172
Disgust	0.130	0.231	1	0.185
Fear	0.159	0.247	1	0.176
Joy	0.130	0.125	–	–
Sadness	0.127	0.174	1	0.141
Surprise	0.060	0.101	1	0.061
Valence	0.159	0.229	2	0.146
Avg. Emo	0.116	0.185	1	0.135
Avg. All	0.122	0.191	1	0.137

Table 4: Performance of expert-trained and non-expert-trained classifiers on test-set. k is the minimum number of non-experts needed to beat an average expert.

For this evaluation we compare the performance of systems trained on expert and non-expert annotations. For each expert annotator we train a system using only the judgments provided by that annotator, and then create a gold standard test set using the average of the responses of the remaining five labelers on that set. In this way we create six independent expert-trained systems and compute the average across their performance, calculated as Pearson correlation to the gold standard; this is reported in the “1-Expert” column of Table 4.

Next we train systems using non-expert labels; for each possible subset of n annotators, for $n \in \{1, 2, \ldots, 10\}$ we train a system, and evaluate by calculating Pearson correlation with the same set of gold standard datasets used in the expert-trained system evaluation. Averaging the results of these studies yields the results in Table 4.

As in Table 2 we calculate the minimum number of non-expert annotations per example k required on average to achieve similar performance to the expert annotations; surprisingly we find that for five of the seven tasks, the average system trained with a single set of non-expert annotations outperforms the average system trained with the labels from a single expert. One possible hypothesis for the cause of this non-intuitive result is that individual labelers (including experts) tend to have a strong bias, and since multiple non-expert labelers may contribute to a single set of non-expert annotations, the annotator diversity within the single set of labels may have the effect of reducing annotator bias and thus increasing system performance.

7 Conclusion

We demonstrate the effectiveness of using Amazon Mechanical Turk for a variety of natural language annotation tasks. Our evaluation of non-expert labeler data vs. expert annotations for five tasks found that for many tasks only a small number of non-expert annotations per item are necessary to equal the performance of an expert annotator. In a detailed study of expert and non-expert agreement for an affect recognition task we find that we require an average of 4 non-expert labels per item in order to emulate expert-level label quality. Finally, we demonstrate significant improvement by controlling for labeler bias.

Acknowledgments

Thanks to Nathanael Chambers, Annie Zaenen, Rada Mihalcea, Qi Su, Panos Ipeirotis, Bob Carpenter, David Vickrey, William Morgan, and Lukas Biewald for useful discussions, and for the generous support of Dolores Labs. This work was supported in part by the Disruptive Technology Office (DTO)’s Advanced Question Answering for Intelligence (AQUAINT) Phase III Program.

References

Paul S. Albert and Lori E. Dodd. 2004. A Cautionary Note on the Robustness of Latent Class Models for Estimating Diagnostic Error without a Gold Standard. Biometrics, Vol. 60 (2004), pp. 427-435.

Collin F. Baker, Charles J. Fillmore and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proc. of COLING-ACL 1998*.

Michele Banko and Eric Brill. 2001. Scaling to Very Very Large Corpora for Natural Language Disambiguation. In *Proc. of ACL-2001*.

Junfu Cai, Wee Sun Lee and Yee Whye Teh. 2007. Improving Word Sense Disambiguation Using Topic Features. In *Proc. of EMNLP-2007* .

Timothy Chklovski and Rada Mihalcea. 2002. Building a sense tagged corpus with Open Mind Word Expert. In *Proc. of the Workshop on "Word Sense Disambiguation: Recent Successes and Future Directions", ACL 2002*.

Timothy Chklovski and Yolanda Gil. 2005. Towards Managing Knowledge Collection from Volunteer Contributors. Proceedings of AAAI Spring Symposium on Knowledge Collection from Volunteer Contributors (KCVC05).

Ido Dagan, Oren Glickman and Bernardo Magnini. 2006. The PASCAL Recognising Textual Entailment Challenge. Machine Learning Challenges. Lecture Notes in Computer Science, Vol. 3944, pp. 177-190, Springer, 2006.

Wisam Dakka and Panagiotis G. Ipeirotis. 2008. Automatic Extraction of Useful Facet Terms from Text Documents. In *Proc. of ICDE-2008*.

A. P. Dawid and A. M. Skene. 1979. Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm. Applied Statistics, Vol. 28, No. 1 (1979), pp. 20-28.

Michael Kaisser and John B. Lowe. 2008. A Research Collection of QuestionAnswer Sentence Pairs. In *Proc. of LREC-2008*.

Michael Kaisser, Marti Hearst, and John B. Lowe. 2008. Evidence for Varying Search Results Summary Lengths. In *Proc. of ACL-2008*.

Phil Katz, Matthew Singleton, Richard Wicentowski. 2007. SWAT-MP: The SemEval-2007 Systems for Task 5 and Task 14. In *Proc. of SemEval-2007*.

Aniket Kittur, Ed H. Chi, and Bongwon Suh. 2008. Crowdsourcing user studies with Mechanical Turk. In *Proc. of CHI-2008*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: the Penn Treebank. Computational Linguistics 19:2, June 1993.

George A. Miller and William G. Charles. 1991. Contextual Correlates of Semantic Similarity. Language and Cognitive Processes, vol. 6, no. 1, pp. 1-28, 1991.

George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunke. 1993. A semantic concordance. In *Proc. of HLT-1993*.

Preslav Nakov. 2008. Paraphrasing Verbs for Noun Compound Interpretation. In *Proc. of the Workshop on Multiword Expressions, LREC-2008*.

Martha Palmer, Dan Gildea, and Paul Kingsbury. 2005. The Proposition Bank: A Corpus Annotated with Semantic Roles. Computational Linguistics, 31:1.

Sameer Pradhan, Edward Loper, Dmitriy Dligach and Martha Palmer. 2007. SemEval-2007 Task-17: English Lexical Sample, SRL and All Words. In *Proc. of SemEval-2007* .

James Pustejovsky, Patrick Hanks, Roser Saur, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro and Marcia Lazo. 2003. The TIMEBANK Corpus. In *Proc. of Corpus Linguistics 2003, 647-656.*

Philip Resnik. 1999. Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language. JAIR, Volume 11, pages 95-130.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual Correlates of Synonymy. Communications of the ACM, 8(10):627–633.

Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis. 2008. Get Another Label? Improving Data Quality and Data Mining Using Multiple, Noisy Labelers. In *Proc. of KDD-2008.*

Push Singh. 2002. The public acquisition of commonsense knowledge. In *Proc. of AAAI Spring Symposium on Acquiring (and Using) Linguistic (and World) Knowledge for Information Access, 2002.*

Alexander Sorokin and David Forsyth. 2008. Utility data annotation with Amazon Mechanical Turk. To appear in *Proc. of First IEEE Workshop on Internet Vision at CVPR, 2008.* See also: `http://vision.cs.uiuc.edu/annotation/`

David G. Stork. 1999. The Open Mind Initiative. IEEE Expert Systems and Their Applications pp. 16-20, May/June 1999.

Carlo Strapparava and Rada Mihalcea. 2007. SemEval-2007 Task 14: Affective Text In *Proc. of SemEval-2007*.

Qi Su, Dmitry Pavlov, Jyh-Herng Chow, and Wendell C. Baker. 2007. Internet-Scale Collection of Human-Reviewed Data. In *Proc. of WWW-2007*.

Luis von Ahn and Laura Dabbish. 2004. Labeling Images with a Computer Game. In ACM Conference on Human Factors in Computing Systems, CHI 2004.

Luis von Ahn, Mihir Kedia and Manuel Blum. 2006. Verbosity: A Game for Collecting Common-Sense Knowledge. In ACM Conference on Human Factors in Computing Systems, CHI Notes 2006.

Ellen Voorhees and Hoa Trang Dang. 2006. Overview of the TREC 2005 question answering track. In *Proc. of TREC-2005*.

Janyce M. Wiebe, Rebecca F. Bruce and Thomas P. O'Hara. 1999. Development and use of a gold-standard data set for subjectivity classifications. In *Proc. of ACL-1999*.

Annie Zaenen. Submitted. Do give a penny for their thoughts. International Journal of Natural Language Engineering (submitted).

HotSpots: Visualizing Edits to a Text

Srinivas Bangalore
AT&T Labs – Research
180 Park Ave
Florham Park, NJ 07932
srini@research.att.com

David Smith
AT&T Labs – Research
180 Park Ave
Florham Park, NJ 07932
dsmith@research.att.com

Abstract

Compared to the telephone, email based customer care is increasingly becoming the preferred channel of communication for corporations and customers. Most email-based customer care management systems provide a method to include template texts in order to reduce the handling time for a customer's email. The text in a template is suitably modified into a response by a customer care agent. In this paper, we present two techniques to improve the effectiveness of a template by providing tools for the template authors. First, we present a tool to track and visualize the edits made by agents to a template which serves as a vital feedback to the template authors. Second, we present a novel method that automatically extracts potential templates from responses authored by agents. These methods are investigated in the context of an email customer care analysis tool that handles over a million emails a year.

1 Introduction

Email based customer care is increasingly becoming the preferred channel of communication for corporations and customers compared to the conventional telephone-based customer care. For customers, email channel offers several advantages – there are no tedious menus to navigate, there is no waiting time to reach an operator, the request can be formulated at the customer's pace and additional material supporting the case can be attached to the email. There is also a record of the service request for the customer unlike the telephone-based customer care. However, there are also limitations of the email channel. The most significant one is that the customer-agent interaction could be drawn out over successive emails spanning over several days as opposed to being resolved in one or two telephone calls. For corporations, the asynchronous nature of email-based customer care offers significant opportunities to reduce operations cost by effective load balancing compared to telephone-based customer care. It is quite common for an email customer care agent to work on several cases simultaneously over a period of a few hours. Email channel also offers higher bandwidth for corporations to send additional information in the form of web links, images and video or audio instructions.

The effectiveness of customer care in the email channel is measured using two competing metrics: Average Handling Time (AHT) and Customer Experience Evaluation (CEE). AHT measures the time taken from when a customer email is opened to the time when the response is sent out. This time is typically averaged over a period of a week or a month for reporting purposes. CEE measures customer satisfaction through a survey of a random subset of customers who have interacted with the email customer care center. These surveys typically involve qualitative and quantitative questions and measure the quality of the interactions along a number of different dimensions. As is the case in many surveys the population responding to such questionnaires is typically small and very often quite biased. We do not use the CEE metric for the work we report in this paper.

As is evident from the definitions of AHT and CEE, it is in the interest of a corporation to minimize AHT while maximizing CEE. In order to reduce AHT, most email customer care systems (Kana, 2008; Genesys, 2008) provide a mechanism for an agent to respond to a customer's email by selecting a predefined template text that can be quickly customized to serve as the response. The template text is usually associated with a problem category it is intended to address and might even be suggested to the agent automatically using classification techniques applied to the customer's email. Once the template is selected, the agent edits the template text to personalize as well as add case specific details as part of composing a response. Each of the text edits contributes to the handling time of the email. Hence, it is in the interest of the template designer to minimize the number of edits of the template in order to lower AHT.

Although most email management systems provide a mechanism to author the template text, there is typically no mechanism to monitor and track how these templates are modified by the agents when they compose a response. This information is vital to the template authors when creating new versions of the templates that reduce the number of edits and consequently reduce AHT.

In this paper, we present two methods for improving the templates in a principled manner. After describing the related work in Section 2, we present a brief description of the email tracking tool we have developed in Section 3. In Section 4, we present a tool called HotSpots that helps visualize the edits being made by the customer care agents to the templates. This tool provides a visual feedback to the template authors and suggests means of improving the template text based on the edits made by agents. In Section 5, we present a new approach to automatically identify emerging templates – texts that are repeatedly created by agents and are similar to each other but distinct from the current template text. We use AHT as the metric to minimize for automatic identification of emerging templates. We discuss some of the issues concerning this work in Section 6 and conclude in Section 7.

2 Related Work

There are few threads of research that are relevant to the work presented in this paper. First, the topic of email response generation in the context of customer care has been investigated by (Coch, 1996; Lapalme and Kosseim, 2003; Zukerman and Marom, 2007). In (Coch, 1996), the authors model multi-sentence generation of response letters to customer complaints in French. The generation model is carefully crafted for the domain using domain-specific rules for conceptual planning, rhetorical relations and surface word order operators. They show that their approach performs better than predefined templates and slightly worse than human generated responses. In (Lapalme and Kosseim, 2003), the authors explore three different approaches based on classification, case-based reasoning and question-answering to compose responses to queries in an email customer care application for the telecommunication industry. The case-based reasoning approach is the most similar to the template approach we follow. In (Zukerman and Marom, 2007), the authors investigate an approach to assembling a response by first predicting the clusters of sentences to be included in the response text and then applying multi-document summarization techniques to collate the representative sentences into a single response. In contrast, in this paper, due to constraints from the deployment environment, we rely on a template-based approach to response generation. We focus on providing tools for investigating how the templates are modified and suggest techniques for evolving more effective templates based on quantitative criteria.

Another thread of relevant research are methods for visualizing texts. There are several methods that have been proposed to provide a visual map of a set of text documents with the focus of illustrating the relatedness of these texts (Card et al., 1999). Using a metric for comparing texts (e.g. n-gram overlap) , the texts are clustered and the resulting clusters are visualized as two or three dimensional color maps. These approaches are useful to depict similarities in a static repository of documents or the return results of a search query. These maps are primarily designed for exploration and navigation through the document space. While the underlying algorithm we use to illustrate the text edits is similar to the one used in text map visualizations, our focus in this paper is to provide a mechanism for template designers to quickly identify the variants of a template sentence created by the agents.

A third thread is in the context of human-assisted machine translation, where a human translator post-edits the output of a machine translation system (Foster et al., 1997; Foster et al., 2002; Och et al., 2003). In order to improve the efficiency of a human translator, the k-best output of a translation system could be displayed as word or phrase choices which are color coded based on the confidence value assigned by the translation model. While the approach we follow is partly motivated by the post-editing paradigm, there are significant differences in the context we apply this approach. In the context of this paper, the template designer is presented a summary of the set of variants created by each agent for each sentence of the template. The task of the template designer is to use this tool to select (or construct) a new variant for the template sentence with the aim of minimizing the need for editing that sentence in future uses of the template.

3 Email Customer Care

Typically, a large email customer care management center receives over 100,000 emails a month. These centers typically use a customer care management system that offer not only logging and tracking of emails but also tools for improving the efficiency of agents responding to emails. Usually, an incoming customer email is categorized into a set of few topics/issues. The categorization might be done automatically based on regular expressions involving keywords in the email or using weighted classifiers that are trained on data. In order for an agent to respond to an incoming email, these systems provide a text box which allows the agent to author a response from scratch. However, most email customer care systems offer the ability to store a prefabricated response (also called *templates*), instead of agents having to author a response from scratch. These templates are typically associated with a problem category or an issue that they are intended to address.

A template helps an agent compose a well-formed response quickly. It contains hints for information that the agent should enter as well as indications of where that information should be entered in the template. The template might also contain helpful information to the customer in addition to legal verbiage that the customer needs to be aware of.

An agent receives a customer email and after comprehending the issues and consulting the customer records in the database, selects one of the predefined set of templates that best addresses the issues raised in the email. Less frequently, she might even select more than one template to compose the response. She then proceeds to edit and personalize the chosen templates to better suit the customer's email. An example of a 'generic' template – not associated with a specific problem category is shown in Figure 1.

Greetings Contact.FirstName,
Thank you for your email in regard to XXXXXXXX.
I will be happy to assist you with your inquiry.
XXX BODY XXX
If I can be of any further assistance,
please reply directly to this email.
Thank you for using our company.
We appreciate your business and continued loyalty.
Regards,
Agent.FirstName

Figure 1: An example of a generic template

The process of selecting an appropriate template that addresses the customer's inquiries could be quite tedious when there are hundreds of templates. Email management systems offer tools that suggest appropriate template to use based on the content of the customer's email. These tools are trained using classification techniques on previous email interactions.

As mentioned earlier, there are two metrics that are typically used to measure the effectiveness and efficiency of email responses. Customers are surveyed after their email interaction to assess their level of satisfaction for the service they received. This is usually called the Customer Experience Evaluation (CEE) and includes an evaluation of the customer's total interaction experience with the corporation, not just the last email interaction. A small subset of customers who had an interaction with the email center is randomly chosen (typically in the order of about 10% of customers) and are invited to take part in the follow-up survey. Typically, only a small percent (about 10%) of the customers who receive these invitations respond to the survey; effectively about 1% of the total emails have customer survey scores.

A second metric that is also used to measure the efficiency of an operation is called the average handling time (AHT) which measures the average of times taken by agents to respond to emails. The handling time includes the time to comprehend the email, the time for database lookup and the time for response composition. It is in the interest of the email customer care operation to minimize AHT and maximize CEE scores.

3.1 Email Customer Care Analysis Tool

We have designed and developed an Email Customer Care Analysis Tool (ECAT) to help analyze the operations of the email care center. It provides an end-to-end view from the activities involved in answering emails to the results of subsequent customer care surveys. In addition, ECAT also provides insights into how the agents are editing templates as well as guides template authors in designing more effective templates.

ECAT is a web-based tool and offers a birds-eye summary of the operations aggregated by region, the template used, and the customer satisfaction survey results. Using this tool, analysts can drill down through a series of views until they are eventually presented with the results of a single survey or a single email interaction.

One of the most useful functions of the tool is that it shows the extent to which agents edit the templates in the process of creating responses to customer emails. The degree to which a template is edited is based on Levenshtein string edit distance metric (Levenshtein, 1966). This metric measures the number of edits (substitution, deletion and insertions) of words that are needed to transform a template into a response. The number of edits is normalized by the number of words in the template. These morphing scores can be viewed for a single email or averaged per agent or per template used. The scores range from 100 to 0, with 100 representing a template which hadn't been edited at all.

The tool also allows the morphing score to be viewed alongside the handling time for an email, in other words the amount of time that the agent spends gathering data and actually composing a response. Handling time is an important metric since it is directly related to cost of operating the email customer care center. The more editing an agent does, the more time they take to respond to a customer. So, the number of templates, their precise wording and the ease with which agents can distinguish them obviously have significant influences on overall handling time.

Beyond the confines of the email centers themselves, the CEE is the most important elements in gauging the effectiveness of the agent. The survey asks customers to rate their overall satisfaction with the email reply to their question. Five is the highest score which equates with 'Extremely Satisfied' while a one equals 'Extremely Dissatisfied.' Customers are also asked to rank the email in terms of it's content, clarity, professionalism and the length of time it took to receive a reply. The customer is also allowed to enter some free text so that they can say how satisfied they were, or not, with how an inquiry or problem was dealt with. Customers can also say whether they called the company using a telephone channel before turning to the email channel.

The survey files, all of which can be accessed in their entirety from within the ECAT tool, also contain information on what templates were used when replying to the customer. They also tell the analyst who the replying agent was and whether this was the first or a subsequent email in communications between the customer and the company.

The ECAT tool juxtaposes this CEE score with the template morphing score to show correlations between customer satisfaction and the degree to which the template had been edited. This data is graphed so that the analyst can immediately see if heavy editing of a template is leading to higher CEE. Heavy editing with a low customer rating could mean that the template is not helping the agent to respond correctly to the customer.

4 HotSpots

We designed the HotSpots tool that provides insights to the template authors on how templates are being edited by the agents when creating responses. It suggests methods for improving the next version of the template so as to reduce edits by agents and hence reduce the handling time for an email. In this section, we discuss the algorithm and the visualization of the information that aids template authors in improving the efficacy of the templates.

The HotSpots algorithm proceeds in two steps as shown in Algorithm 1. The first step creates an alignment between the template string and the re-

Algorithm 1 Compute HotSpots for a template T given a response set $\mathcal{R}$

```
1: $EdEv = \phi$
2: $T = s_1 s_2 \dots s_n$
3: $T_s = \{s_i | 1 \le i \le n\}$
4: $R_s = \{r_i^j | R_j \in \mathcal{R}, R_j = r_1^j r_2^j \dots r_{m_j}^j, 1 \le i \le m_j\}$
5: $Index : \{T_s \cup R_s\} \rightarrow \mathbb{I}$
6: $T_{in} = Index(s_1) Index(s_2) \dots Index(s_n)$
7: for all $R \in \mathcal{R}$ do
8:    $R = r_1 r_2 \dots r_{n_R}$
9:    $R_{in} = Index(r_1) Index(r_2) \dots Index(r_{n_R})$
      // compute distance with sentences as tokens and return the alignment and score
10:   $(alignment, score) = IndDist(T_{in}, R_{in})$
      // for each of the sentences in T, update its map
11:   for all $s_i \in T$ do
12:      $in = Index(s_i)$
13:      if $(s_i, \epsilon) \in alignment$ then
14:         $EdEv[in].map = EdEv[in].map \cup \{*delete*\}$
15:      else // $(s_i, r_j) \in alignment$
16:         $EdEv[in].map = EdEv[in].map \cup \{r_j\}$
17:      end if
18:   end for
19: end for
    // Cluster the response sentences aligned for each template sentence
20: for all $s_i \in T$ do
21:    $in = Index(s_i)$
22:    $Cl = KmedianCl(EdEv[in].map, ncl)$
23: end for
```

Algorithm 2 KmedianCl: Compute k centroids for a set of strings S using k-median clustering algorithm

```
1: $c_s = \phi$ // centroid of string $s$'s cluster
2: $ce_i = \phi$ // centroid of cluster $i$
3: $numcl = 0$ // number of cluster created so far
4: $Cl_i = \phi$ // members of cluster $i$
5: while $(numcl \le k) \wedge (numcl \le |S|)$ do
6:    if $numcl = 0$ then
7:       $ce_0 = \underset{c \in S}{argmin} \sum_{s \in S} Dist(c, s)$
8:    else // select the string ($s$) that is farthest from its centroid ($c_s$)
9:       $ce_{numcl} = \underset{s \in S}{argmax}\, Dist(c_s, s)$
10:   end if
      // Move strings to the closest cluster and compute centroids until the set of centroids don't change
11:   repeat
12:      for all $s \in S$ do
13:         $i^* = \underset{0 \le i \le numcl}{argmin}\, Dist(ce_i, s)$
14:         $c_s = ce_{i^*}$
15:         $Cl_{i^*} = Cl_{i^*} \cup \{s\}$
            // Computed the closest cluster centroid $ce_i$ to $s$.
16:      end for
         // Recompute the cluster centroids $ce_i$
17:      for all $i$ such that $0 \le i \le numcl$ do
18:         $ce_i = \underset{c \in Cl_i}{argmin} \sum_{s \in Cl_i} Dist(c, s)$
19:      end for
20:   until set of centroids does not change
21:   $numcl = numcl + 1$ // new cluster added
22: end while
```

sponse string. For the purposes of this paper, we consider the alignments between the template text and the response text with sentences as tokens instead of a word-based alignment. The rationale for this tokenization is that for template developers the visualization of the edits is expected to be more meaningful when aggregated at the sentence level rather than at the word level. In the second step, using the sentence-level alignment we compute the edit events (insertion, deletion and substitution) of the template sentences in order to create the response. All the edits events associated with a template sentence are then clustered into k clusters and the centroids of the k clusters are displayed as the potential changes to that sentence. We next describe these two steps in detail as illustrated in Algorithm 1 and Algorithm 2.

Given a set of responses $\mathcal{R}$ that agents create using a template T, Algorithm 1 proceeds as follows. Each of the sentences in the template and the set of responses are mapped into an integer index (Line 1). The template T and each of the responses in R are split into sentences and mapped into index sequences (Line 6 and Line 9). The alignment between the two index strings is computed in Line 10. This is a dynamic programming algorithm similar to computing Levenshtein distance between two strings, except the cost function used to compute the match between tokens is as shown below.

From the alignment that maps s_i to r_j, we collect the set of response sentences associated with each template sentence (Line 13-16). These sentences are then clustered using k-median clustering method (illustrated in Algorithm 2) in Line 22.

In Algorithm 2, we illustrate the method of clustering we use to summarize the set of sentences we have collected for each template sentence after the alignment step. The algorithm is similar to the k-means algorithm (Duda et al., 2001), however, given that we are clustering strings instead of real numbers (as is typical in applications of k-means), we restrict the centroid of a cluster to be one of the members of the set being clustered, hence the name k-median algorithm (Martnez-Hinarejos et al., 2003). The distance function to measure the closeness of two strings is instantiated to be an n-gram overlap between the two strings.[1]

The algorithm iterates over three steps until the data is partitioned into k clusters (Line 5). The first step (Lines 6-10) is the initialization of a centroid for a new cluster. Initially when the data is not partitioned into any cluster, the median string of the data set is used as the initial centroid. For subsequent iterations, the farthest point from all the centroids computed thus far is used as the centroid for the new cluster. In the second step (Lines 11-16), each member of the data set is assigned to the nearest cluster based on its distance to that cluster's centroid. Finally, in the third step (Lines 17-20), the cluster centroids are recomputed based on the new cluster memberships. Steps two and three are repeated until there are no changes in the cluster memberships and cluster centroids. This completes the introduction of a new cluster for the data.

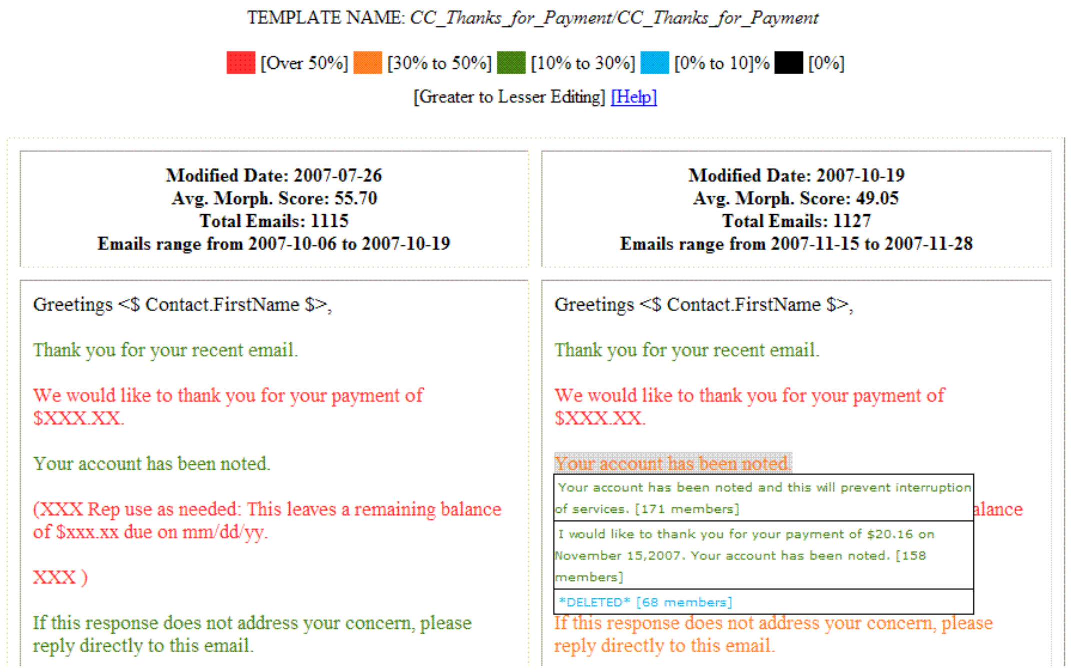
For the purposes of our task, we use up to a four-gram overlap to measure distance between two strings and use $k = 5$ for clustering the data.

4.1 Visualizing the HotSpots

The HotSpots page was created within the ECAT tool to surgically dissect the way in which templates were being morphed. For a given template, as shown in Figure 2, the analyst is presented with a copy of the texts from the current and previous versions of that template. Each sentence in the two versions of the template are color coded to show how frequently the agents have changed that sentence. This involved running the HotSpots algorithm against approximately 1,000 emails per template version. A sentence that is colored red is one that was changed in over 50% of the emails that were responded to using that template. An orange sentence is one that was edited in between 30% and 50%, green is between 10% to 30% and blue is between 0% and 10%. The more often a sentence is edited the 'hotter' the color.

The analyst can see the typical substitutions for a sentence by hovering the mouse over that sentence. The typical sentences computed as the centroids of the clusters created using Algorithm 2 are themselves color coded using the same identification sys-

[1]We have also experimented with a symmetric version of Levenshtein distance, but we prefer the n-gram overlap score due to its linear run time complexity.


TEMPLATE NAME: *CC_Thanks_for_Payment/CC_Thanks_for_Payment*

🟥 [Over 50%] 🟧 [30% to 50%] 🟩 [10% to 30%] 🟦 [0% to 10]% ⬛ [0%]

[Greater to Lesser Editing] [Help]

Modified Date: 2007-07-26 **Avg. Morph. Score: 55.70** **Total Emails: 1115** **Emails range from 2007-10-06 to 2007-10-19**	**Modified Date: 2007-10-19** **Avg. Morph. Score: 49.05** **Total Emails: 1127** **Emails range from 2007-11-15 to 2007-11-28**
Greetings <$ Contact.FirstName $>, Thank you for your recent email. We would like to thank you for your payment of $XXX.XX. Your account has been noted. (XXX Rep use as needed: This leaves a remaining balance of $xxx.xx due on mm/dd/yy. XXX) If this response does not address your concern, please reply directly to this email.	Greetings <$ Contact.FirstName $>, Thank you for your recent email. We would like to thank you for your payment of $XXX.XX. Your account has been noted. Your account has been noted and this will prevent interruption of services. [171 members] I would like to thank you for your payment of $20.16 on November 15,2007. Your account has been noted. [158 members] *DELETED* [68 members] alance If this response does not address your concern, please reply directly to this email.

Figure 2: Example of two versions of a template and the edit score (Avg. Morph. Score) and centroids associated with each sentence of the template.

tem. A typical sentence that occurred in over 50% of the emails is colored red. A typical sentence that occurred in 30% to 50% of the emails was orange and so on.

In seeing the two versions side by side, the analyst can visually inspect the agents' edits on the current version of a template relative to the previous version. If the previous version of the template is a 'hotter' document (with more red sentences), it means that the changes made to the template by the author had led to less editing by agents thus speeding up the process of creating a customer response. If the current template looks hotter, it suggests that the changes made to the template were increasing the agents' edits and probably the email handling time.

5 Automatic Extraction of Potential Templates

The goal of the template author is to minimize the number of edits done to a template and thus indirectly lowering the handling time for an email. In the preceding section, we discussed a tool that aids the template authors to identify sentences where changes are most often made by agents to a template. This information could be used by the template authors to create a new version of the template that achieve the goal.

In this section, we investigate a technique that automatically identifies a possible template with the potential of directly minimizing the average handling time for an email. We use the set of responses created by the agents using a given template and select one of the responses to be generalized and stored as a new template. The response to be converted into a template is chosen so as to directly minimize the average handling time. In essence, we seek to partition the set of responses R generated from template T into two clusters R_1 and R_2. These clusters have centroids T (current template) and T' (new template) such that constraint shown in 1 holds.

$$\begin{aligned}(\forall_{r \in R_1} AHT(T,r) < AHT(T',r)) \wedge \\ (\forall_{r \in R_2} AHT(T',r) < AHT(T,r))\end{aligned} \qquad (1)$$

Now, the quantity $AHT(T,r)$ is logged as part of the email management system and corresponds to the time taken to respond to a customer's email.[2]

[2]Although typically this time includes the time to look up a

Cluster	Number of members	Centroid (Template/Response)
1	1799	GREETINGSPHR, Thank you for your recent email. On behalf of the company, I would like to extend my sincere apology for the problems you encountered when (XXX over key with appropriate response XXX). It is our goal to provide excellent customer service, and I am sorry that we did not meet that objective. Your input is very valuable, and we will take your feedback into consideration. Regards, Agent.FirstName
2	206	GREETINGSPHR, Thank you for letting me know that you've been unable to send an online order to upgrade your NAMEDENTITY service. Please accept my apologies for any problems this issue may have caused you. You're a highly valued customer. I understand your concerns and I'll be happy to address them. I am investigating this issue. I have already made a personal commitment to email you tomorrow, with the resolution. Thank you for your patience and for choosing the company. We appreciate your business and continued loyalty. Sincerely, Agent.FirstName

Table 1: Result of clustering responses using the AHT model as the distance metric.

However, we do not have access to $AHT(T', r)$ for any $T' \neq T$. We propose a model to estimate this in the next section.

5.1 Modeling Average Handling Time

We model AHT as a linear combination of several factors which we believe would influence the handling time for an email. These factors include the length in words of the customer's input email ($inplen$), the length in words of the template ($templatelen$), the length in words of the response ($resplen$), the total number of edits between the template and the response ($edit$), the normalized edit score ($nedit$), the number of individual events of the edit distance – substitution (sub), insertion (ins), deletion (del) and identity (id), the number of block (contiguous) substitution ($blksub$), block insertion ($blkins$) and block deletion ($blkdel$). Using these independent variables, we fit a linear regression model using the AHT values for 6175 responses created from one particular template (say G). The result of the regression fit is shown in Equation 2 and the data and error statistics are shown in Table 2. It must be noted that the coefficients for the variables are not necessarily reflective of the importance of the variables, since they compensate for the different ranges in variable values. We have also tried several different regression fits with fewer variables, but find that this fit gives us the best correlation with the data.

$$\begin{aligned} \hat{AHT} = {} & 0.5314 * inplen - 2.7648 * templatelen \\ & +1.9982 * resplen - 0.5822 * edit \\ & +2900.5242 * nedit \\ & +4.7499 * id - 1.6647 * del \\ & -1.6021 * ins + 26.6704 * blksub \\ & -15.239 * blkins + 24.3931 * blkdel \\ & -261.6627 \end{aligned} \quad (2)$$

Mean AHT	675.74 seconds
Median AHT	543 seconds
Mode AHT	366 seconds
Standard Deviation	487.72 seconds
Correlation coefficient	0.3822
Mean absolute error	320.2 seconds
Root mean squared error	450.64 seconds
Total Number of Instances	6175

Table 2: Data statistics and the goodness of the regression model for 6175 AHT data points.

Based on the goodness statistics of the regression fit, it is clear the AHT model could be improved further. However, we acknowledge that AHT does not depend solely on the editing of a template to a

the customer's account etc., we assume that time is quite similar for all responses created from the same template.

response but involves several other components including the user interface, the complexity of customer's email, the database retrieval to access the customer's account and so forth.

Nevertheless, we use this model to cluster a new set of 2005 responses originating from the same template (G), as shown in Equation 1. Using the k-median clustering as described earlier, we partition the responses into two clusters. We restrict the first cluster centroid to be the template and search for the best centroid for the second cluster. The results are shown in Table 1. The centroid for cluster 1 with 1799 members is the template itself while the centroid for cluster 2 with 206 members is a response that could be suitably generalized to serve as a template. The overall AHT for the 2005 responses using the template was 989.2 seconds, while the average AHT for the members of cluster 1 and 2 was 971.9 seconds and 1140 seconds, indicating that the template had to be edited considerably to create the members of cluster 2.

6 Discussion

For the purposes of this paper, it is assumed that AHT is the same as or correlates well with the time to compose a response for an email. However, in most cases the email care agent might have to perform several verification, validation, and problem resolution phases by consulting the specifics of a customer account before formulating and composing a response. The time taken for each of these phases typically varies depending on the customer's account and the problem category. Nevertheless, we assume that the times for these phases is mostly a constant for a given problem category, and hence the results presented in this paper need to be interpreted on a per problem category basis.

A second limitation of the approach presented in this paper is that the metric used to measure the similarity between strings (n-gram overlap) is only a crude approximation of an ideal semantic similarity metric. There are however other similarity metrics (e.g. BLEU (Papineni et al., 2002)) which could be used equally well. The purpose of this paper is to illustrate the possibility of analysis of responses using one particular instantiation of the similarity metric.

In spite of the several directions that this work can be improved, the system and algorithms described in this paper have been deployed in an operational customer care center. The qualitative feedback we have received are extremely positive and analysts have greatly improved the efficiency of the operation using this tool.

7 Conclusions

In this paper, we have presented two approaches that help template authors in designing effective templates for email customer care agents. In the first approach, we have presented details of a graphical tool that provides vital feedback to the template authors on how their templates are being modified by agents when creating responses. The template authors can accommodate this information when designing the next version of the template. We also presented a novel technique for identifying responses that can potentially serve as templates and reduce AHT. Towards this end, we discussed a method to model AHT based on the characteristics of the customer's email, the template text and the response text.

8 Acknowledgments

We would like to thanks Mazin Gilbert, Junlan Feng, Narendra Gupta and Wenling Hsu for the discussions during the course of this work. We also thank the members who generously offered to their support to provide us with data used in this study without which this work would not have been possible. We thank the anonymous reviewers for their useful suggestions in improving the quality of this paper.

References

S.K. Card, J. Mackinlay, and B. Shneiderman. 1999. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann.

J. Coch. 1996. Evaluating and comparing three text-production techniques. In *Proceedings of Coling-96*, pages 249–254, Copenhagen, Denmark.

R.O. Duda, P.E. Hart, and D.G. Stork. 2001. *Pattern Classification*. Wiley, New York.

G. Foster, P. Isabelle, and P. Plamondon. 1997. Target text mediated interactive machine translation. *Machine Translation*, 12(1):175–194.

G. Foster, P. Langlais, and G. Lampalme. 2002. User-friendly text prediction for translators. In *EMNLP-02*, pages 46–51, Philadelphia, USA.

Genesys. 2008. http://www.genesys.com. Genesys Corporation.

Kana. 2008. http://www.kana.com. Kana Corporation.

G. Lapalme and L. Kosseim. 2003. Mercure: Towards an automatic e-mail follow-up system. *IEEE Computational Intelligence Bulletin*, 2(1):14–18.

V.I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertion and reversals. *Soviet Physics Doklady*, 10:707–710.

C.D. Martnez-Hinarejos, A. Juan, and F. Casacuberta. 2003. Generalized k-medians clustering for strings. *Lecture Notes in Computer Science*, 2652/2003:502–509.

F.J. Och, R. Zens, and H. Ney. 2003. Efficient search for interactive statistical machine translation. In *EACL-03*.

K. Papineni, S. Roukos, T. Ward, and W.J. Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association of Computational Linguistics*, pages 313–318, Philadelphia, PA, July.

I. Zukerman and Y. Marom. 2007. Evaluation of a large-scale email response system. In *Proceedings of IJ-CAI07*, Hyderabad, India.

Who is Who and What is What: Experiments in Cross-Document Co-Reference

Alex Baron
BBN Technologies
10 Moulton Street
Cambridge, MA 02138
abaron@bbn.com

Marjorie Freedman
BBN Technologies
10 Moulton Street
Cambridge, MA 02138
mfreedma@bbn.com

Abstract

This paper describes a language-independent, scalable system for both challenges of cross-document co-reference: name variation and entity disambiguation. We provide system results from the ACE 2008 evaluation in both English and Arabic. Our English system's accuracy is 8.4% relative better than an exact match baseline (and 14.2% relative better over entities mentioned in more than one document). Unlike previous evaluations, ACE 2008 evaluated both name variation and entity disambiguation over naturally occurring named mentions. An information extraction engine finds document entities in text. We describe how our architecture designed for the 10K document ACE task is scalable to an even larger corpus. Our cross-document approach uses the names of entities to find an initial set of document entities that could refer to the same real world entity and then uses an agglomerative clustering algorithm to disambiguate the potentially co-referent document entities. We analyze how different aspects of our system affect performance using ablation studies over the English evaluation set. In addition to evaluating cross-document co-reference performance, we used the results of the cross-document system to improve the accuracy of within-document extraction, and measured the impact in the ACE 2008 within-document evaluation.

1 Introduction

Cross-document entity co-reference is the problem of identifying whether mentions from different documents refer to the same or distinct entities. There are two principal challenges: the same entity can be referred to by more than one name string (e.g. Mahmoud Abbas and Abu Mazen) and the same name string can be shared by more than one entity (e.g. John Smith). Algorithms for solving the cross-document co-reference problem are necessary for systems that build knowledge bases from text, question answering systems, and watch list applications.

There are several challenges in evaluating and developing systems for the cross-document co-reference task. (1) The annotation process required for evaluation and for training is expensive; an annotator must cluster a large number of entities across a large number of documents. The annotator must read the context around each instance of an entity to make reliable judgments. (2) On randomly selected text, a baseline of exact string match will do quite well, making it difficult to evaluate progress. (3) For a machine, there can easily be a scalability challenge since the system must cluster a large number of entities.

Because of the annotation challenges, many previous studies in cross-document co-reference have focused on only the entity disambiguation problem (where one can use string retrieval to collect many documents that contain same name); or have used artificially ambiguated data.

Section 2 describes related work; section 3 introduces ACE, where the work was evaluated; section 4 describes the underlying information extraction engine; sections 5 and 6 address the challenges of coping with name variation and disambiguating entities; sections 7, 8, and 9 present empirical results, improvement of entity extraction within documents using cross-document coreference, and a difference in performance on person versus organization entities. Section 10 discusses the scalability challenge. Section 11 concludes.

2 Related Work

Person disambiguation given a person name string. Bagga and Baldwin (1998b) produced one of the first works in cross-document co-reference. Their work presented a vector space model for the problem of entity disambiguation, clustering 197 articles that contained the name 'John Smith'.

Participants in the 2007 Sem-Eval Web People Search(WEPS) task clustered 100-document sets based on which person a name string of interest referenced. WEPS document sets were collected by selecting the top 100 web search results to queries about a name string (Artiles, et al., 2007).

Mann and Yarowsky (2003) and Gooi and Allan (2004) used artificially ambiguous data to allow for much larger experiments in clustering documents around a known person of interest.

Clustering different variants of the same name. Lloyd et. al (2006) use a combination of 'morphological similarity' and 'contextual similarity' to cluster name variants that refer to the same entity.

Clustering and disambiguation. The John Hopkins 2007 Summer Workshop produced a cross-document annotated version of the ACE 2005 corpus (18K document entities, 599 documents) consisting of 5 entity types (Day, et. al, 2007). There was little ambiguity or variation in the corpus. Participants demonstrated that disambiguation improvements could be achieved with a Metropolis-Hastings clustering algorithm. The study assumed human markup of document-level entities.

Our work. The work reported in this paper addresses both entity clustering and name variation for both persons and organizations in a corpus of 10K naturally occurring documents selected to be far richer than the ACE 2005 data by NIST and LDC. We investigated a new approach in both English and Arabic, and evaluated on document-level entities detected by information extraction.

3 ACE Evaluation

NIST's ACE evaluation measures system performance on a predetermined set of entities, relations, and events. For the 2008 global entity detection and recognition task (GEDR)[1], system performance was measured on named instances of person and organization entities. The GEDR task was run over both English and Arabic documents. Participants processed over 10K documents for each language. References were produced for about 400 documents per language (NIST, 2008). The evaluation set included documents from several genres over a 10 year time period. Document counts are provided in Table 1. This evaluation differed from previous community cross-document coreference evaluations in that it (a) covered both organizations and people; (b) required processing a relatively large data set; (c) evaluated entity disambiguation and name variation simultaneously; and (d) measured cross-document co-reference over system-detected document-level entities and mentions.

	English	**Arabic**
broadcast conversation	8	38
broadcast news	72	19
meeting	18	---
newswire	237	314
telephone	18	12
usenet	15	15
weblog	47	14

Table 1: Documents per genre in ACE2008 test set

The evaluation set was selected to include interesting cases for cross-document co-reference (e.g cases with spelling variation and entities with shared names). This is necessary because annotation is difficult to produce and naturally sampled data has a high percentage of entities resolvable with string match. The selection techniques were unknown to ACE participants.

4 Extraction System Overview

Our cross-document co-reference system relies on SERIF, a state-of-the-art information extraction (IE) system (Ramshaw, et. al, 2001) for document-level information extraction. The IE system uses statistically trained models to detect and classify mentions, link mentions into entities, and detect and classify relations and events. English and Arabic SERIF share the same general models, although there are differences in the specific features used by the models. Arabic SERIF does not perform event detection. While Arabic SERIF does

[1] NIST's evaluation of cross-document co-reference.

make use of some morphological features, the cross-document co-reference system, which focused specifically on entity names, does not use these features.

Figure 1 and Figure 2 illustrate the architecture and algorithms of the cross-document co-reference system respectively. Our system separately addresses two aspects of the cross-document co-reference problem: name variation (Section 5) and entity disambiguation (Section 6). This leads to a scalable solution as described in Section 10.

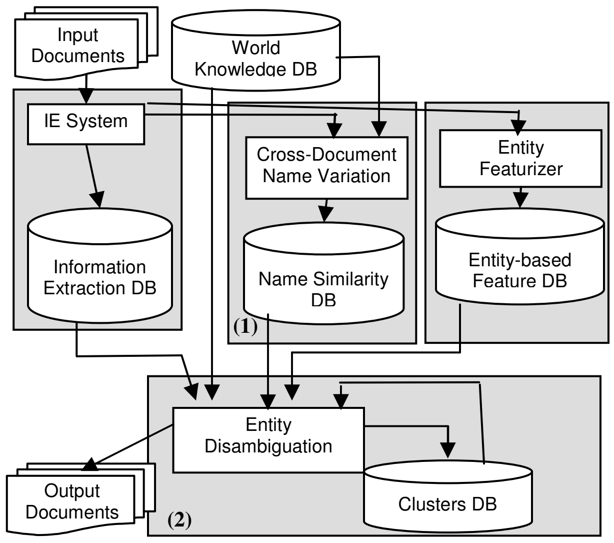


Figure 1: Cross-document Co-reference Architechure

The features used by the cross-document co-reference system can be divided into four classes: World Knowledge (W), String Similarity (S), Predictions about Document Context (C), and Metadata (M). Name variation (V) features operate over unique corpus name strings. Entity disambiguation features (D) operate over document-level entity instances. During disambiguation, the agglomerative clustering algorithm merges two clusters when conditions based on the features are met. For example, two clusters are merged when they share at least half the frequently occurring nouns that describe an entity (e.g. president). As shown in Table 2, features from the same class were often used in both variation and disambiguation. All classes of features were used in both English and Arabic. Because very little training data was available, both the name variation system and the disambiguation system use manually tuned heuristics to combine the features. Tuning was done using the ACE2008 pilot data (LDC, 2008b), documents from the SemEval WEPS task (Artiles, et al., 2007), and some internally annotated documents. Internal annotation was similar in style to the WEPS annotation and did not include full ACE annotation. Annotators simply clustered documents based on potentially confusing entities. Internal annotation was done for ~100 names in both English and Arabic.

Feature Class	**Stage**	**Class**
Wikipedia knowledge	D, V	W
Web-mined aliases	V	W
Word-based similarity	D, V	S
Character-based similarity	V	S
Translation dictionaries	V	S
Corpus Mined Aliases	D, V	C
SERIF extraction	D,V	C
Predicted Document Topics	D	C
Metadata (source, date, etc.)	D	M

Table 2: Features for Cross-Document Co-Reference

5 Name Variation

The name variation component (Block 1 of Figure 1) collects all name strings that appear in the document set and provides a measure of similarity between each pair of name strings.[2] Regions (A) and (B) of Figure 2 illustrate the input and output of the name variation component.

This component was initially developed for question answering applications, where when asked the question *'Who is George Bush?'* relevant answers can refer to both George W and George HW (the question is ambiguous). However when asked '*Who leads al Qaeda?'* the QA system must be able to identify spelling variants for the name al Qaeda. For the cross-document co-reference problem, separating the name variation component from the disambiguation component improves the scalability of the system (described in Section 10).

The name variation component makes use of a variety of features including web-mined alias lists, aliases mined from the corpus (e.g *'John aka J'*), statistics about the relations and co-reference decisions predicted by SERIF, character-based edit distance, and token subset trees. The token subset trees algorithm measures similarity using word overlap by building tree-like structures from the unique corpus names based on overlapping tokens. Translation dictionaries (pulled from machine

[2] For the majority of pairs, this similarity score will be 0.

translation training and cross-language links in Wikipedia) account for names that have a canonical form in one language but may appear in many forms in another language.

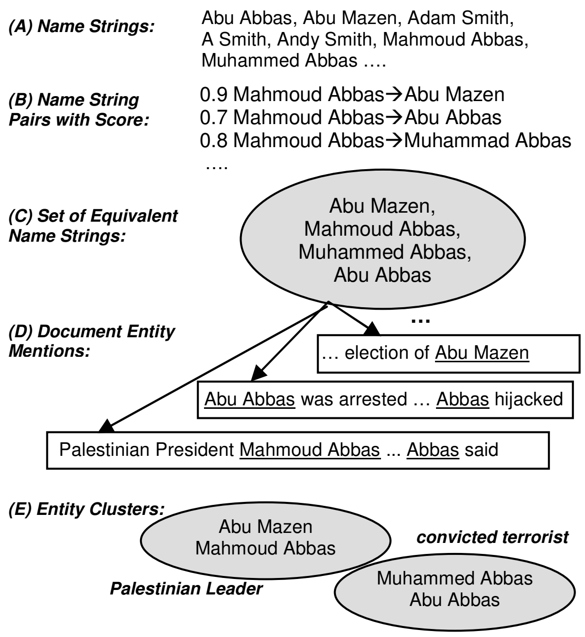

(A) Name Strings:
Abu Abbas, Abu Mazen, Adam Smith, A Smith, Andy Smith, Mahmoud Abbas, Muhammed Abbas ….
(B) Name String Pairs with Score:
0.9 Mahmoud Abbas→Abu Mazen
0.7 Mahmoud Abbas→Abu Abbas
0.8 Mahmoud Abbas→Muhammad Abbas
….
(C) Set of Equivalent Name Strings:
Abu Mazen, Mahmoud Abbas, Muhammed Abbas, Abu Abbas
(D) Document Entity Mentions:
…
… election of Abu Mazen
Abu Abbas was arrested … Abbas hijacked
Palestinian President Mahmoud Abbas ... Abbas said
(E) Entity Clusters:
Abu Mazen Mahmoud Abbas
Palestinian Leader
convicted terrorist
Muhammed Abbas Abu Abbas

Figure 2: Cross-document Co-reference Process

The features are combined with hand-tuned weights resulting in a unidirectional similarity score for each pair of names. The similarity between two name strings is also influenced by the similarity between the contexts in which the two names appear (for example the modifiers or titles that precede a name). This information allows the system to be more lenient with edit distance when the strings appear in a highly similar context, for example increasing the similarity score between *'Iranian President Ahmadinejad'* and *'Iranian President Nejad.'*

6 Entity Disambiguation

We use a complete link agglomerative clustering algorithm for entity disambiguation. To make agglomerative clustering feasible over a 10K document corpus, rather than clustering all document-level entities together, we run agglomerative clustering over subsets of the corpus entities. For each name string, we select the set of names that the variation component chose as valid variants. In Figure 2 region C, we have selected Mahmoud Abbas and 3 variants.

We then run a three stage agglomerative clustering algorithm over the set of document entities that include any of the name string variants or the original name. Figure 2 region D illustrates three document-level entities.

The name variation links are not transitive, and therefore a name string can be associated with more than one clustering instance. Furthermore document-level entities can include more than one name string. However once a document-level entity has been clustered, it remains linked to entities that were a part of that initial clustering. Because of this, the order in which the algorithm selects name strings is important. We sort the name strings so that those names about which we have the most information and believe are less likely to be ambiguous are clustered first. Name strings that are more ambiguous or about which less information is available are clustered later.

The clustering procedure starts by initializing singleton clusters for each document entity, except those document entities that have already participated in an agglomerative clustering process. For those entities that have already been clustered, the clustering algorithm retrieves the existing clusters.

The merging decisions are based on the similarity between two clusters as calculated through feature matches. Many features are designed to capture the context of the document in which entities appear. These features include the document topics (as predicted by the unsupervised topic detection system (Sista, et al., 2002), the publication date and source of a document, and the other names that appear in the document (as predicted by SERIF). Other features are designed to provide information about the specific context in which an entity appears for example: the noun phrases that refer to an entity and the relationships and events in which an entity participates (as predicted by SERIF). Finally some features, such as the uniqueness of a name in Wikipedia are designed to provide the disambiguation component with world knowledge about the entity. Since each cluster represents a global entity, as clusters grow through merges, the features associated with the clusters expand. For example, the set of associated document topics the global entity participates in grows.

While we have experimented with statistically learning the threshold for merging, because of the small amount of available training data, this threshold was set manually for the evaluation.

Clustering over these subsets of similar strings has the additional benefit of limiting the number of global decisions that are affected by a mistake in the within-document entity linking. For example, if in one document, the system linked *Hillary Clinton* to *Bill Clinton*; assuming that the two names are not chosen as similar variants, we are likely to end up with a cluster made largely of mentions of *Hillary* with one spurious mention of *Bill* and a separate cluster that contains all other mentions of *Bill.* In this situation, an agglomerative clustering algorithm that linked over the full set of document-level entities is more likely to be led astray and create a single *'Bill and Hillary'* entity.

7 Experimental Results

Table 3 and Table 4 include preliminary ACE results[3] for the highest, lowest, and average system in the local and cross-document tasks respectively. While a single participant could submit more than one entry, these numbers reflect only the primary submissions. The ACE scorer maps system produced entities to reference entities and produces several metrics. For the within-document task, metrics include ACE Value, B3, and a variant of B3 weighted to reflect ACE value weightings. For the cross-document task, the B3 metric is replaced with F (NIST, 2008). ACE value has traditionally been the official metric of the ACE evaluation. It puts a higher cost on certain classes of entities (e.g. people are more important than facilities), certain classes of mentions (e.g. names are more important than pronouns), and penalizes systems for mistakes in type and subtype detection as well as linking mistakes. Assigning a mention to the wrong entity is very costly in terms of value score. If the mention is a name, a system is penalized 1.0 for the missed mention and an additional 0.75 for a mention false alarm. We will report ACE Value and value weighted B3/F. Scores on the local task are not directly comparable to scores on the global task. The local entity detection and recognition task (LEDR) includes entity detection for five (rather than two) classes of entities and includes pronoun and nominal (e.g. 'the group') mentions in addition to names.

[3] Results in this paper use v2.1 of the references and v17 of the ACE scorer. Final results will be posted to http://www.nist.gov/speech/tests/ace/2008/

	English		**Arabic**	
	Val	B3Val	Val	B3Val
Top	52.6	71.5	43.6	69.1
Average	-53.3	50.0	17.3	47.6
Low[4]	-269.1	25.8	-9.1	26.1
BBN-A-edropt	52.1	71.5	43.0	68.9
BBN-B-st-mg	52.6	71.5	43.6	69.1
BBN-B-st-mg-fix[5]	57.2	77.4	44.6	71.3

Table 3: ACE 2008 Within-Document Results (LEDR)

	English		**Arabic**	
	Val	FVal	Val	FVal
Top	53.0	73.8	28.2	58.7
Average	21.1	59.1	24.7	56.8
Low	-64.1	31.6	21.2	54.8
BBN-B-med	53.0	73.8	28.2	58.7
BBN-B-low	53.2	73.8	28.7	59.3
BBN-B-med-fix[5]	61.7	77	31.4	60.1

Table 4: ACE 2008 Cross-Document Results (GEDR)

Our cross-document co-reference system used BBN-A-edropt as input. BBN-B-st-mg is the result of using cross-document co-reference to improve local results (Section 9). For cross-document co-reference, our primary submission, BBN-B-med, was slightly outperformed by an alternate system BBN-B-low. The two submissions differed only in a parameter setting for the topic detection system (BBN-B-low requires more documents to predict a 'topic'). BBN-A-st-mg-fix and BBN-B-med-fix are the result of post-processing the BBN output to account for a discrepancy between the training and evaluation material.[5]

In addition to releasing results, NIST also released the references. Table 5 includes the ACE score for our submitted English system and the score when the system was run over only the 415 documents with references. The system performs slightly better when operating over the full document set. This suggests that the system is using information from the corpus even when it is not directly scored.

[4] There was a swap in rank between metrics, so the low numbers reflect two different systems.

[5] There were discrepancies between the ACE evaluation and training material with respect to the portions of text that should be processed. Therefore our initial system included a number of spurious entities. NIST has accepted revised output that removes these entities. Experiments in this paper reflect the corrected system.

	FVal
10K documents processed (415 scored) (BBN-B-med-fix)	77
Only 415 documents processed	76.3

Table 5: Full English System ACE Evaluation Results

We have run a series of ablation experiments over the 415 files in the English test set to evaluate the effectiveness of different feature classes. These experiments were run using only the annotated files (and not the full 10K document set). We ran two simple baselines. The first baseline ('No Link') does not perform any cross-document co-reference, all document entities are independent global entities. The second baseline ('Exact Match') links document-level entities using exact string match. We ran 6 variations of our system:

- Configuration 1 is the most limited system. It uses topics and IE system output for disambiguation, and aliases mined from the documents for the name variation component.
- Configuration 2 includes Configuration 1 features with the addition of string similarity (edit distance, token subset trees) algorithms for the name variation stage.
- Configuration 3 includes Configuration 2 features and adds context-based features (e.g. titles and premodifiers) for name variation.
- Configuration 4 adds information from document metadata to the disambiguation component.
- Configuration 5 adds web-mined information (alias lists, Wikipedia, etc.) to both the variation and disambiguation components. This is the configuration that was used for our NIST submission.
- Configuration 5a is identical to Configuration 5 except that the string-based edit distance was removed from the name variation component.

As noted previously, the ACE collection was selected to include challenging entities. The selection criteria of the corpus (which are not known by ACE participants) can affect the importance of features. For example, a corpus that included very few transliterated names would make less use of features based on edit distance.

Figure 3 and Figure 4 show performance (with value weighted F) on the eight conditions over system predicted within-document extraction and reference within-document extraction respectively. Figure 3 also includes configuration 5 run over all 10K documents. We provide two sets of results. The first evaluates system performance over all entities. The relatively high score of the 'No Link' baseline indicates that a high percentage of the document-level entities in the corpus are only mentioned in one document. The second set of numbers measures system performance on those entities appearing in more than one reference document. While this metric does not give a complete picture of the cross-document co-reference task (sometimes a singleton entity must be disambiguated from a large entity that shares the same name); it does provide useful insights given the frequency of singleton entities.

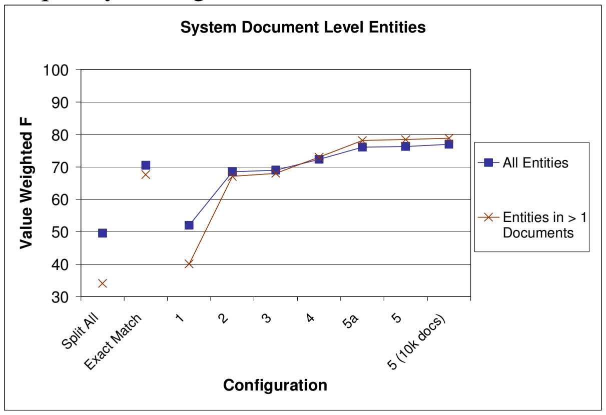


Figure 3: Performance on System Document Entities

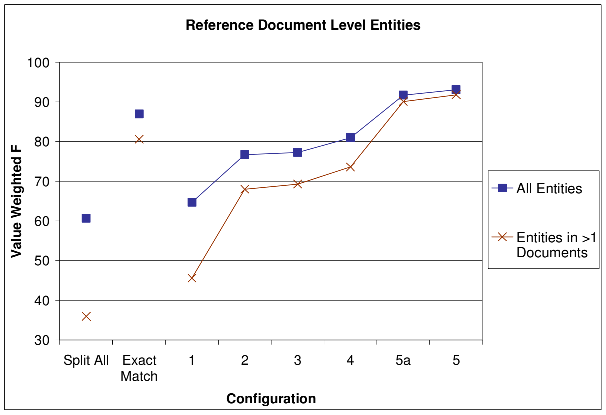


Figure 4: Performance on Perfect Document Entities

Overall system performance improved as features were added. Configuration 1, which disambiguated entities with a small set of features, performed worse than a more aggressive exact string match strategy. The nature of our agglomerative clustering algorithm leads to entity merges only when there is sufficient evidence for the merge. The relatively high performance of the exact match strategy suggests that in the ACE corpus, most entities that shared a name string referred to the same entity, and therefore aggressive merging leads to better performance. As additional features are added, our system becomes more confident and merges more document-level entities.

With the addition of string similarity measures (Configuration 2) our system outperforms the exact match baseline. The submitted results on system entities (Configuration 5) provide a 8.4% relative reduction in error over the exact match baseline. If scored only on entities that occur in more than one document, Configuration 5 gives a 14.2% relative redution in error over the exact match baseline.

The context based features (Configuration 3) allow for more aggressive edit-distance-based name variation when two name strings frequently occur in the same context. In Configuration 3, *'Sheik Hassan Nasrallah'* was a valid variant of *'Hassan Nasrallah'* because both name strings were commonly preceded by '*Hezbollah leader*'. Similarly, *'Dick Cheney'* became a valid variant of '*Richard Bruce Cheney'* because both names were preceded by '*vice president*'. In Configuration 2 the entities included in both sets of name strings had remained unmerged because the strings were not considered valid variants. With the addition of contextual information (Configuration 3), the clustering algorithm created a single global entity. For the '*Dick Cheney'* cluster, this was correct. *'Sheik Hassan Nassrallah'* was a more complex instance, in some cases linking was correct, in others it was not.

The impact of the metadata features (Configuration 4) was both positive and negative. An article about the '*Arab League Secretary General Amru Moussa'* was published on the same day in the same source as an article about '*Intifada Fatah movement leader Abu Moussa*'. With the addition of metadata features, these two distinct global entities were merged. However, the addition of metadata features correctly led to the merging of three instances of the name '*Peter'* in ABC news text (all referring ABC's Peter Jennings).

Web-mined information (Configuration 5) provides several variation and disambiguation features. As we observed, the exact match baseline has fairly high accuracy but is obviously also too aggressive of a strategy. However, for certain very famous global entities, any reference to the name (especially in corpora made of primarily news text) is likely to be a reference to a single global entity. Because these people/organizations are famous, and commonly mentioned, many of the topic and extraction based features will provide insufficient evidence for merging. The same famous person will be mentioned in many different contexts. We use Wikipedia as a resource for such entities. If a name is unambiguous in Wikipedia, then we merge all instances of this name string. In the evaluation corpus, this led to the merging of many different instances of '*Osama Bin Laden'* into a single entity. Web-mined information is also a resource for aliases and acronyms. These alias lists, allowed us to merge '*Abu Muktar'* with '*Khadafi Montanio'* and '*National Liberation Army'* with *'ELN'*.

Interestingly, removing the string edit distance algorithm (System 5a), is a slight improvement over System 5. Initial error analysis has shown that while the string edit distance algorithm did improve accuracy on some entities (e.g linking '*Sam Alito'* with *'Sam Elito'* and linking '*Andres Pastrana'* with '*Andreas Pastrana*'*);* in other cases, the algorithm *allowed* the system to overlink two entities, for example linking *'Megawati Soekarnoputri'* and her sister '*Rachmawati Sukarnoputri'*.

8 Improving Document-Level Extraction with Global Information

In addition to evaluating the cross-document system performance on the GEDR task, we ran a preliminary set of experiments using the cross-document co-reference system to improve within-document extraction. Global output modified within-document extraction in two ways.

First, the cross-document co-reference system was used to modify the within-document system's subtype classification. In addition to evaluating entity links and type classification, the ACE task measures subtype classification. For example, for organization entities, systems distinguish between Media and Entertainment organizations. The IE system uses all mentions in a given entity to assign a subtype. The cross-document co-reference system has merged several document-level entities, and therefore has even more information with which to assign subtypes. The cross-document system also has access to a set of manual labels that have been assigned to Wikipedia categories.

Secondly, we used the cross-document co-reference system's linking decisions to merge within-document entities. If the cross-document co-reference system merged two entities in the same document, then those entities were merged in the within-document output.

Table 6 includes results for our within-document IE system, the IE system with improved subtypes, and the IE system with improved subtypes and merged entities.

	B3Val	Val
Local	77.3	56.7
+ Subtypes	77.3	56.9
+ Merge	77.4	57.2

Table 6: Within-document Results

While these preliminary experiments yield relatively small improvements in accuracy, an analysis of the system's output suggests that the merging approach is quite promising. The output that has been corrected with global merges includes the linking entities with 'World Knowledge' acronyms (e.g. linking *'FARC'* with *'Armed Revolutionary Forces of Colombia');* linking entities despite document-level extraction mistakes (e.g. '*Lady Thatcher*' with '*Margaret Thatcher');* and linking entities despite spelling mistakes in a document (e.g linking *'Avenajado'* with *'Robert Aventajado').* However, as we have already seen, the cross-document co-reference system does make mistakes and these mistakes can propagate to the within-document output.

In particular, we have noticed that the cross-document system has a tendency to link person names with the same last name when both names appear in a single document. As we think about the set of features used for entity disambiguation, we can see why this would be true. These names may have enough similarity to be considered equivalent names. Because they appear in the same document, they will have the same publication date, document source, and document topics. Adjusting the cross-document system to either use a slightly different approach to cluster document-level entities from the same document or at the very least to be more conservative in applying merges that are the result primarily of document metadata and context to the within-document output could improve accuracy.

9 Effect of LEDR on GEDR

Unlike previous evaluations of cross-document co-reference performance, the ACE 2008 evaluation included both person and organization entities. We have noticed that the performance of the cross-document co-reference system on organizations lags behind the performance of the system on people. In contrast, for LEDR, the extraction system's performance is quite similar between the two entity classes. Furthermore, the difference between global organization and person accuracy in the GEDR is smaller when the GEDR task performed with perfect document-level extraction. Scores are shown in Table 7. These differences suggest that part of the reason for the low performance on organizations in GEDR is within-document accuracy.

	LEDR		GEDR-System		GEDR-Perfect	
	B3Val	Val	FVal	Val	FVal	Val
Org	75.1	51.7	67.8	45.9	91.5	84.0
Per	76.2	52.9	83.2	71.4	94.3	89.5

Table 7: Performance on ORG and PER Entities

The LEDR task evaluates names, nominals, and pronouns. GEDR, however only evaluates over name strings. To see if this was a part of the difference in accuracy, we removed all pronoun and nominal mentions from both the IE system's local output and the reference set. As shown in Table 8, the gap in performance between organizations and people is much larger in this setting.

	LEDR- Name Only	
	B3Val	Val
ORG	82.6	83.0
PER	90.1	90.4

Table 8: Local Performance on Name Only Task

Because the GEDR task focuses exclusively on names and excludes nominals and pronouns, mistakes in mention type labeling (e.g. labeling a name as a nominal) become misses and false alarms rather than type substitutions. As the task is currently defined, type substitutions are much less costly than a missing or false alarm entity.

Intuitively, correctly labeling the name of a person as a name and not a nominal is simple. The distinction for organizations may be fuzzier. For example the string 'the US Department of Justice' could conceivably contain one name, two names, or a name and a nominal. The ACE guidelines (LDC, 2008a) suggest that this distinction can be difficult to make, and in fact have a lengthy set of rules for classifying such cases. However, these rules can seem unintuitive, and may be difficult for machines to learn. For example 'Justice Department' is not a name but 'Department of Justice' is. In some sense, this is an artificial distinction enforced by the task definition, but the accuracy numbers suggest that the distinction has a negative effect on system evaluation.

10 Scalability

One of the challenges for systems participating in the ACE task was the need to process a relatively large document set (10K documents). In question answering applications, our name variation algorithms have been applied to even larger corpora (up to 1M documents). There are two factors that make our solution scalable.

First, much of the name variation work is highly parallelizable. Most of the time spent in this algorithm is spent in the name string edit distance calculation. This is also the only algorithm in the name variation component that scales quadratically with the number of name strings. However, each calculation is independent, and could be done simultaneously (with enough machines). For the 10K document set, we ran this algorithm on one machine, but when working with larger document sets, these computations were run in parallel.

Second, the disambiguation algorithm clusters subsets of document-level entities, rather than running the clustering over all entities in the document set. In the English ACE corpus, the IE system found more than 135K document-level entities that were candidates for global entity resolution. There were 62,516 unique name strings each of which was used to initialize an agglomerative clustering instance. As described in Section 6, a document entity is only clustered one time. Consequently, 36% of these clustering instances are 'skipped' because they contain only already clustered document entities. Even the largest clustering instance contained only 1.4% of the document-level entities.

The vast majority of agglomerative clustering instances disambiguated a small number of document-level entities and ran quickly. 99.7% of the agglomerative clustering runs took less than 1 second. 99.9% took 90 seconds or less.

A small number of clustering instances included a large number of document entities, and took significant time. The largest clustering instance, initialized with the name string '*Xinhua,*' contained 1848 document-level entities (1.4% of the document-level entities in the corpus). This instance took 2.6 hours (27% of the total time spent running agglomerative clustering). Another frequent entity '*George Bush*' took 1.2 hours.

As described in Section 6, the clustering procedure can combine unresolved document-level entities into existing global entities. For large cluster sets (e.g entities referred to by the string '*Xinhua*'), speed would be improved by running many smaller clustering instances on subsets of the document-level entities and then merging the results.

11 Conclusions and Future Work

We have presented a cross-document co-reference clustering algorithm for linking entities across a corpus of documents that

- addresses both the challenges of name variation and entity disambiguation.
- is language-independent,
- is scalable

As measured in ACE 2008, for English our system produced an .8.4% relative reduction in error over a baseline that used exact match of name strings. When measured on only entities that appeared in more than one document, the system gave a 14.2% relative reduction in error. For the Arabic task, our system produced a 7% reduction in error over exact match (12.4% when scored over entities that appear in more than one document). We have shown how a variety of features are important for addressing different aspects of the cross-document co-reference problem. Our current features are merged with hand-tuned weights. As additional development data becomes available, we believe it would be feasible to statistically learn the weights. With statistically learned weights, a larger feature set could improve accuracy even further.

Global information from the cross-document co-reference system improved within-document information extraction. This suggests both that a document-level IE system operating over a large corpus text can improve its accuracy with information that it learns from the corpus; and also that integrating an IE system more closely with a source of world knowledge (e.g. a knowledge base) could improve extraction accuracy.

Acknowledgements
This work was supported by the United States Department of Homeland Security. Elizabeth Boschee and Ralph Weischedel provided useful insights during this work.

References

Artiles, Javier, Julio Gonzalo. & Felisa Verdejo.. 2005. A Testbed for People Searching Strategies. In *the WWW. SIGIR 2005 Conference*. Salvador, Brazil.

Artiles, Javier, Julio Gonzalo. & Satochi Sekine.. 2007. The SemEval-2007 WePS Evaluation: Establishing a benchmark for the Web People Search Task. *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*, pages 64–69, Prague, Czech.

Bagga, Amit & Breck Baldwin. 1998a. Algorithms for Scoring Coreference Chains. In *Proceedings of the Linguistic Coreference Workshop at the First International Conference on Language Resources and Evaluation (LREC'98),* pages 563-566.

Bagga, Amit & Breck Baldwin. 1998b. Entity-Based Cross-Document Coreferencing Using the Vector Space Model. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (COLING-ACL'98*), pages 79-85.

Day, David.,Jason Duncan, Claudio Guiliano, Rob Hall, Janet Hitzeman,Su Jian, Paul McNamee, Gideon Mann, Stanley Yong & Mike Wick. 2007. CDC Features. *Johns Hopkins Summer Workshop on Cross-Document Entity Disambiguation*. http://www.clsp.jhu.edu/ws2007/groups/elerfed/documents/fullCDED.ppt

Gooi, Chung Heong & James Allan. 2004. Cross-document coreference on a large scale corpus. *In Human Language Technology Conf. North American Chapter Association for Computational Linguistics*, pages 9–16, Boston, Massachusetts, USA.

Lloyd, Levon., Andrew Mehler & Steven Skiena 2006. Identifying Co-referential Names Across Large Corpora. *Combinatorial Pattern Matching*. 2006, pages 12-23, Barcelona, Spain.

Linguistic Data Consortium 2008a. ACE (Automatic Content Extraction) English Annotation Guidelines for Entities Version 6.6 2008.06.13. . Linguistic Data Consortium, Philadelphia. http://projects.ldc.upenn.edu/ace/docs/English-Entities-Guidelines_v6.6.pdf

Linguistic Data Consortium, 2008b. ACE 2008 XDOC Pilot Data V2.1. LDC2007E64. Linguistic Data Consortium, Philadelphia.

Mann, Gideon S. & Yarowsky, David. 2003. Unsupervised Personal Name Disambiguation In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL*, pages 33-40.

NIST Speech Group. 2008. The ACE 2008 evaluation plan: Assessment of Detection and Recognition of Entities and Relations Within and Across Documents. http://www.nist.gov/speech/tests/ace/2008/doc/ace08-evalplan.v1.2d.pdf

Ramshaw, Lance, E. Boschee, S. Bratus, S. Miller, R. Stone, R. Weischedel and A. Zamanian: "Experiments in Multi-Modal Automatic Content Extraction"; in Proc. of HLT-01, San Diego, CA, 2001.

Sista, S, R. Schwartz, T. Leek, and J. Makhoul. An Algorithm for Unsupervised Topic Discovery from Broadcast News Stories. In Proceedings of ACM HLT, San Diego, CA, 2002.

Arabic Named Entity Recognition using Optimized Feature Sets

Yassine Benajiba[•] **Mona Diab**[◇] **Paolo Rosso**[•]

[•]Natural Language Engineering Lab.,
Dept. de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
{ybenajiba,prosso}@dsic.upv.es
[◇]Center of Computational Learning Systems
Columbia University
mdiab@cs.columbia.edu

Abstract

The Named Entity Recognition (NER) task has been garnering significant attention in NLP as it helps improve the performance of many natural language processing applications. In this paper, we investigate the impact of using different sets of features in two discriminative machine learning frameworks, namely, Support Vector Machines and Conditional Random Fields using Arabic data. We explore lexical, contextual and morphological features on eight standardized data-sets of different genres. We measure the impact of the different features in isolation, rank them according to their impact for each named entity class and incrementally combine them in order to infer the optimal machine learning approach and feature set. Our system yields a performance of $F_{\beta=1}$-measure=83.5 on ACE 2003 Broadcast News data.

1 Introduction

Named Entity Recognition (NER) is the process by which named entities are identified and classified in an open-domain text. NER is one of the most important sub-tasks in Information Extraction. Thanks to standard evaluation test beds such as the Automatic Content Extraction (ACE)[1], the task of NER has garnered significant attention within the natural language processing (NLP) community. ACE has facilitated evaluation for different languages creating standardized test sets and evaluation metrics. NER systems are typically enabling sub-tasks within large NLP systems. The quality of the NER system has a direct impact on the quality of the overall NLP system. Evidence abound in the literature in areas such as Question Answering, Machine Translation, and Information Retrieval (Babych and Hartley, 2003; Ferrández et al., 2004; Toda and Kataoka, 2005). The most prominent NER systems approach the problem as a classification task: identifying the named entities (NE) in the text and then classifying them according to a set of designed features into one of a predefined set of classes (Bender et al., 2003). The number of classes differ depending on the data set. To our knowledge, to date, the approach is always to model the problem with a single set of features for all the classes simultaneously. This research, diverges from this view. We recognize that different classes are sensitive to differing features. Hence, in this study, we aspire to discover the optimum feature set per NE class. We approach the NER task from a multi-classification perspective. We create a classifier for each NE class independently based on an optimal feature set, then combine the different classifiers for a global NER system. For creating the different classifiers per class, we adopt two discriminative approaches: Support Vector Machines (SVM)(Vapnik, 1995), and Conditional Random Fields (CRF)(Lafferty et al., 2001). We comprehensively investigate many sets of features for each class of NEs: contextual, lexical, morphological and shallow syntactic features. We explore the feature sets in isolation first. Then, we employ the Fuzzy Borda Voting Scheme (FBVS) (García Lapresta and Martínez Panero, 2002) in order to rank the features according to their perfor-

[1]http://www.nist.gov/speech/tests/ace/2004/doc/ace04-evalplan-v7.pdf

mance per class. The incremental approach to feature selection leads to an interpretable system where we have a better understanding of the resulting errors. The paper is structured as follows: Section 2 gives a general overview of the state-of-the-art NER approaches with a particular emphasis on Arabic NER; Section 3 describes relevant characteristics of the Arabic language illustrating the challenges posed to NER; in Section 4.1 we describe the Support Vector Machines and Conditional Random Fields Modeling approaches. We discuss details about our feature-set in 4.2 and describe the Fuzzy Borda Voting Scheme in Section 4.3. Section 5 describes the experiments and shows the results obtained; Withing Section 5, Section 5.1 gives details about the data-sets which we use; finally, we discuss the results and some of our insights in Section 6 and draw some conclusions in 7.

2 Related Work

To date, the most successful language independent approaches to English NER are systems that employ Maximum Entropy (ME) techniques in a supervised setting (Bender et al., 2003).

(Tran et al., 2007) show that using a Support Vector Machine (SVM) approach outperforms ($F_{\beta=1}$=87.75) using CRF ($F_{\beta=1}$=86.48) on the NER task in Vietnamese. For Arabic NER, (Benajiba et al., 2007) show that using a basic ME approach yields $F_{\beta=1}$=55.23. Then they followed up with further work in (Benajiba and Rosso, 2007), where they model the problem as a two step classification approach applying ME, separating the NE boundary detection from the NE classification. That modification showed an improvement in performance yielding an $F_{\beta=1}$=65.91. None of these studies included Arabic specific features, all the features used were language independent. In a later study, (Benajiba and Rosso, 2008) report using lexical and morphological features in a single step model using CRF which resulted in significant improvement over state of the art to date for Arabic NER, yielding $F_{\beta=1}$=79.21. However, the data that was used in these evaluation sets were not standard sets. Most recently, (Farber et al., 2004) have explored using a structured perceptron based model that employs Arabic morphological features. Their system benefits from the basic POS tag (15 tags) information and the corresponding capitalization information on the gloss corresponding to the Arabic word. Exploiting this information yields a significant improvement in recall of 7% and an overall $F_{\beta=1}$=69.6 on the ACE2005 data set. The authors note the lack of improvement in the system's performance when using other Arabic morphological information.

3 Arabic in the context of NER

The Arabic language is a language of significant interest in the NLP community mainly due to its political and economic significance, but also due to its interesting characteristics. Arabic is a Semitic language. It is known for its templatic morphology where words are made up of roots, patterns, and affixes. Clitics agglutinate to words. For instance, the surface word وبحسناتهم *wbHsnAthm*[2] 'and by their virtues[fem.]', can be split into the conjunction *w* 'and', preposition *b* 'by', the stem *HsnAt* 'virtues [fem.]', and possessive pronoun *hm* 'their'.

With respect to the NER task, Arabic poses several major challenges:

Absence of capital letters in the orthography: English like many other Latin script based languages has a specific marker in the orthography, namely capitalization of the initial letter, indicating that a word or sequence of words is a named entity. Arabic has no such special signal rendering the detection of NEs more challenging.

Absence of short vowels: The absence of short vowels renders the lexical items a lot more ambiguous than in other languages exacerbating the homography problem. The average polysemy for surface unvowelized words in Arabic is 12 possible vowelized forms and when the inflections are removed the average is 4 possible vowelized forms.[3] For instance, words such as براد *brAd* can be read both as 'refrigerator' or 'Brad',respectively, where the former is a common noun and the latter is an NE.

[2]We use the Buckwalter transliteration scheme to show romanized Arabic (Buckwalter, 2002).

[3]It is worth noting that each vowelized form could still be ambiguous as in the English homograph/homophone 'bank' case.

The Arabic language is highly inflectional: As we mentioned earlier, Arabic language uses an agglutinative strategy to form surface tokens. As seen in the example above, a surface Arabic word may be translated as a phrase in English. Consequently, the Arabic data in its raw surface form (from a statistical viewpoint) is much more sparse which decreases the efficiency of training significantly.

4 Our Approach

We approach the problem of NER from a per NE class based perspective. The intuition is that features that are discriminative for one NE class might not be for another class. In the process, we decide on an optimal set of features for each NE class. Finally we combine the different classifiers to create a global NER system. Hence, we identify a set of features for NER and proceed to investigate them individually. Then we use an automatic ranking system to pick the optimal set of features per NE class. To that end, we use the Fuzzy Borda Voting Scheme (FBVS). We employ two discriminative classification techniques: Support Vector Machines (SVM) and Conditional Random Fields (CRF). Even though some previous studies seem to point to the superiority of SVM over CRF for NER (Tran et al., 2007), it is hard to draw a definitive conclusion since their assessment was based on comparing the average F-measure.[4] Moreover, the best system to date on Arabic NER reports results using CRF (Benajiba and Rosso, 2008). We adopt an IOB2 annotation scheme for classification. For each NE class, we have two types of class labels: B-Class, marking the beginning of a Class chunk, and I-Class marking the inside of a class chunk. Finally, we mark words not participating in an NE as O, meaning they are outside some NE class label.

4.1 SVM and CRF

SVM approach is based on Neural Networks (Vapnik, 1995). The goal is to find, in the training phase, the best decision function which allows us to obtain the class c for each set of features f. SVM are robust to noise and have powerful generalization ability, especially in the presence of a large number of features. Moreover, SVM have been used successfully in many NLP areas of research in general (Diab et al., 2007), and for the NER task in particular (Tran et al., 2007). We use a sequence model *Yamcha toolkit*,[5] which is defined over SVM.

CRF are a generalization of Hidden Markov Models oriented toward segmenting and labeling sequence data (Lafferty et al., 2001). CRF are undirected graphical models. During the training phase the conditional likelihood of the classes are maximized. The training is discriminative. They have been used successfully for Arabic NER (see section 2). We have used *CRF++*[6] for our experiments.

4.2 Our Feature Sets

One of the most challenging aspects in machine learning approaches to NLP problems is deciding on the optimal feature sets. In this work, we investigate a large space of features which are characterized as follows:

Contextual (CXT): defined as a window of $+/-$ n tokens from the NE of interest

Lexical (LEX_i): defined as the lexical orthographic nature of the tokens in the text. It is a representation of the character n-grams in a token. We define the lexical features focusing on the first three and last three character n-grams in a token. Accordingly, for a token $C_1C_2C_3...C_{n-1}Cn$, then the lexical features for this token are LEX_1=C_1, LEX_2=C_1C_2, LEX_3=$C_1C_2C_3$, $LEX4$=C_n, $LEX_5 = C_{n-1}C_n$, $LEX_6 = C_{n-2}C_{n-1}C_n$.

Gazetteers (GAZ): These include hand-crafted dictionaries/gazetteers listing predefined NEs. We use three gazetteers for person names, locations and organization names.[7] We semi-automatically enriched the location gazetteer using the Arabic Wikipedia[8] as well as other web sources. This enrichment consisted of: (i) taking the page labeled "*Countries of the world*" (دول العالم, dwl AlEAlm) as a starting point to crawl into Wikipedia and retrieve location names; (ii) we automatically filter the data removing stop words; (iii) finally, the resulting

[4]The authors did not report any per class comparison between SVM and CRF.

[5]http://chasen.org/∼taku/software/yamcha/

[6]http://crfpp.sourceforge.net/

[7]http://www.dsic.upv.es/∼ybenajiba

[8]http://ar.wikipedia.org

list goes through a manual validation step to ensure quality. On the training and test data, we tag only the entities which exist entirely in the gazetteer, e.g. if the entity ‘United States of America’ exists in our gazetteer, we would not tag ‘United States’ on the data as a location. Exception is made for person names. We augment our dictionary by converting the multiword names to their singleton counterparts in addition to keeping the multiword names in the list. We tag them on the evaluation data separately. Accordingly, the name ‘Bill Clinton’ and ‘Michael Johnson’ as two entries in our dictionary, are further broken down to ‘Bill’, ‘Clinton’, ‘Michael’, ‘Johnson’. The intuition is that the system will be able to identify names such as ‘Bill Johnson’ and ‘Clinton’ as person names. This is always true for person names, however this assumption does not hold for location or organization names.

Part-Of-Speech (POS) tags and Base Phrase Chunks (BPC): To derive part of speech tags (POS) and base phrase chunks (BPC) for Arabic, we employ the AMIRA-1.0 system[9] described in (Diab et al., 2007). The POS tagger has a reported accuracy of 96.2% (25 tags) and the BPC system performs at a reported $F_{\beta=1}$=96.33%, assuming gold tokenization and POS tagging.

Nationality (NAT): The input is checked against a manually created list of nationalities.

Morphological features (MORPH): This feature set is based on exploiting the characteristic rich morphological features of the Arabic language. We rely on the MADA system for morphological disambiguation (Habash and Rambow, 2005), to extract relevant morphological information. MADA disambiguates words along 14 different morphological dimensions. It typically operates on untokenized texts (surface words as they naturally occur), hence, several of the features indicate whether there are clitics of different types. We use MADA for the preprocessing step of clitic tokenization (which addresses one of the challenges we note in Section 3, namely the impact different morphological surface forms have on sparseness). Recognizing the varying importance of the different morphological features and heeding the reported MADA performance per feature, we carefully engineered the choice of the relevant morphological features and their associated value representations. We selected 5 morphological features to include in this study.

[9]http://www1.cs.columbia.edu/∼mdiab/

1. Aspect (M_{ASP}) : In Arabic, a verb maybe imperfective, perfective or imperative. However since none of the NEs is verbal, we decided to turn this feature into a binary feature, namely indicating if a token is marked for Aspect (APP, for applicable) or not (NA, for not applicable).

2. Person (M_{PER}) : In Arabic, verbs, nouns, and pronouns typically indicate person information. The possible values are *first, second* or *third* person. Again, similar to *aspect*, the applicability of this feature to the NEs is more relevant than the actual value of *first* versus *second*, etc. Hence, we converted the values to APP and NA, where APP applies if the person feature is rendered as *first, second* or *third*.

3. Definiteness (M_{DEF}) : MADA indicates whether a token is definite or not. All the NEs by definition are definite. The possible values are DEF, INDEF or NA.

4. Gender (M_{GEN}) : All nominals in Arabic bear *gender* information. According to MADA, the possible values for this feature are masculine (MASC), feminine (FEM), and neuter (or not applicable NA), which is the case where gender is not applicable for instance in some of the closed class tokens such as prepositions, or in the case of verbs. We use the three possible values MASC, FEM and NA, for this feature. The intuition is that since we are using a sequence model, we are likely to see agreement in *gender* information in participants in the same NE.

5. Number (M_{NUM}) : For almost all the tokens categories (verbs, nouns, adjectives, etc.) MADA provides the grammatical *number*. In Arabic, the possible values are singular (SG), dual (DU) and plural (PL). The correlation of the SG value with most of the NEs classes is very high. Heeding the underlying agreement of words in Arabic when they are part of the same NE, the values for this feature are SG, DU, PL and NA (for cases where number is not applicable such as closed class function words).

Corresponding English Capitalization (CAP): MADA provides the English translation for the words it morphologically disambiguates as it is based on an underlying bilingual lexicon. The intuition is that if the translation begins with a capital letter, then it is most probably a NE. This feature is an attempt to overcome the lack of capitalization for NEs in Arabic (see Section 3). This is similar to the *GlossCAP* feature used in (Farber et al., 2004).

4.3 Fuzzy Borda Voting Scheme

Fuzzy Borda Voting Scheme (FBVS) is useful when several possible candidates (c_n) are ranked by different experts (e_m) and we need to infer a single ranking (García Lapresta and Martínez Panero, 2002). It is based on the Borda count method which was introduced by Jean-Charles de Borda in 1770. In FBVS, each expert provides the ranking of the candidates with a weight[10] (w_n^m) assigned to each of them. Thereafter, for each expert e_i, we generate a square matrix such as $e_i = (r_{1,1}^i \dots r_{n,n}^i)$ where:

$$r_{j,k}^i = \frac{w_j^i}{w_j^i + w_k^i} \tag{1}$$

Given each expert matrix, we calculate for each row $r'^i_j = \sum_k r_{j,k}^i;\ r_{j,k}^i > \alpha$ where α is a certain threshold. Accordingly, for each candidate, we sum up the weights obtained from the different experts in order to obtain a final weight for each candidate ($r''^j = \sum_i r'^i_j$). Finally, we rank them according to r''^j. In our experiments, the candidates we rank are the features. The FBVS ranking is calculated per ML technique and class of NEs across all the data sets according to the features' performances $\mathrm{F}_{\beta=1}$, i.e. the weights. The $\mathrm{F}_{\beta=1}$ ranges from $0-1$. We use $\alpha = 0.5$, thereby taking into consideration only the features which have shown a significant difference in performance.

5 Experiments and Results

5.1 Data

We report the results of our experiments on the standard sets of ACE 2003, ACE 2004 and ACE 2005 data sets.[11] The ACE data (see Table 1) is annotated for many tasks: Entity Detection and Tracking (EDT), Relation Detection and Recognition (RDR), Event Detection and Recognition (EDR). All the data sets comprise *Broadcast News* (BN) and *Newswire* (NW) genres. ACE 2004 includes an additional NW data set from the Arabic TreeBank (ATB). ACE 2005 includes a different genre of *Weblogs* (WL).

$Corpus$	$genre$	$Size_{train}$	$Size_{dev}$	$Size_{test}$
ACE 2003	BN	12.41k	4.12k	5.63k
	NW	23.85k	9.5k	9.1k
ACE 2004	BN	45.68k	14.44k	14.81k
	NW	45.66k	15.2k	16.9k
	ATB	19.04k	6.16k	6.08k
ACE 2005	BN	18.54k	5k	8.4k
	NW	40.26k	12.5k	13.83k
	WL	13.7k	6.2k	6.4

Table 1: Statistics of ACE 2003, 2004 and 2005 data

We create a dev, test and train set for each of the collections. Table 1 gives the relevant statistics. It is worth noting that the standard training sets have 4 folds that are typically used for training. We used one of the folds as dev data for tuning purposes, rendering our training data less for our experiments. For data preprocessing, we remove all annotations which are not oriented to the EDR task. Also, we remove all the 'nominal' and 'pronominal' mentions of the entities and keep only the 'named' ones. Hence, all the listed characteristics for this corpus pertain to the portions of the data that are relevant to NER only. The ACE 2003 data defines four different NE classes: Person (e.g. Albert Einstein), Geographical and Political Entities (GPE) (e.g. Kazakhistan), Organization (e.g. Google Co.) and Facility (e.g. the White House). Whereas in ACE 2004 and 2005, two NE classes are added to the ACE 2003 tag-set: Vehicles (e.g. Rotterdam Ship) and Weapons (e.g. Kalashnikof). In order to overcome the sparseness issues resulting , we clitic tokenize the text using the MADA system. We use the ATB style clitic tokenization standard. Finally, we convert the data from the ACE format into the IOB2 annotation scheme (Tjong Kim Sang and De Meudler, 2003).

5.2 Experimentation

Our objective is to find the optimum set of features per NE class and then combine the outcome in a

[10]weights are not required for classical Borda count.
[11]http://www.nist.gov/speech/tests/ace/

global NER system for Arabic. We set the context window to be of size $-1/+1$ for all the experiments, as it empirically yields the best performance. We use the CoNLL evaluation metrics of precision, recall, and $F_{\beta=1}$ measures. The CoNLL metrics are geared to the chunk level yielding results as they pertain to the entire NE (Tjong Kim Sang and De Meudler, 2003). Our experiments are presented as follows:

1. Training per individual NE class: We train for an individual class by turning off the other annotations for the other classes in the training set. We experimented with two settings: 1. Setting all the other NE classes to O, similar to non-NE words, thereby yielding a 3-way classification, namely, B-NE and I-NE for the class of interest, and O for the rest including the rest of the NEs and other words and punctuation; 2. The second setting discriminated between the other NE classes that are not of interest and the rest of the words. The intuition in this case is that NE class words will naturally behave differently than the rest of the words in the data. Thereby, this setting yields a 4-way classification: B-NE and I-NE for class of interest, NE for the other NE classes, and O for the other words and punctuation in the data. In order to contrast the 3-way vs the 4-way classification, we run experiments and evaluate using the ACE 2003 data set with no features apart from 'CXT' and 'current word' using SVM. Table 2 illustrates the yielded results. For all

Class	*Num(classes)*	*BN genre*	*NW genre*
GPE	3	76.72	79.88
	4	**76.88**	**80.99**
PER	3	64.34	42.93
	4	**67.56**	**44.43**
ORG	3	41.73	25.24
	4	**46.02**	**37.97**
FAC	3	**23.33**	15.3
	4	**23.33**	**18.12**

Table 2: $F_{\beta=1}$ Results using 3-way vs. 4-way class annotations using SVM

the NE classes we note that the 4-way classification yields the best results. Moreover, we counted the number of 'conflicts' obtained for each NE classification. A 'conflict' arises when the same token is classified as a different NE class by more than one classification system. Our findings are summarized as follows:

(i). 3 classes: 16 conflicts (8 conflicts in BN and 8 in NW). 10 of these conflicts are between GPE and PER, and 6 of them are between GPE and ORG.

(ii). 4 classes: 10 conflicts (3 conflicts in BN and 7 in NW). 9 of these conflicts are between GPE and ORG, and only one of them is between GPE and FAC.

An example of a conflict observed using the 3-way classification that disappeared when we apply the 4-way classification is in the following sentence: نشرت صحيفة واشنطن تايمس تقريرا *n$rt SHyfp WA$nTn tAyms tqryrA*, which is translated as 'The Washington Times newspaper published a report'. When trained using a 3-way classifier, 'Washington' is assigned the tag GPE by the GPE classifier system and as an ORG by the ORG classifier system. However, when trained using the 4-way classifier, this conflict is resolved as an ORG in the ORG classifier system and an NE in the GPE classifier system. Thereby confirming our intuition that a 4-way classification is better suited for the individual NE classification systems. Accordingly, for the rest of the experiments in this paper reporting on individual NE classifiers systems, we use a 4-way classification approach.

2. Measuring the impact of Individual features per class : An experiment is run for each fold of the data. We train on data annotated for one NE class, one Machine Learning (ML) method (i.e. SVM or CRF), and one feature. For each experiment we use the tuning set for evaluation, i.e. obtaining the $F_{\beta=1}$ performance value.

3. FBVS Ranking : After obtaining the F-measures for all the individual features on all the data genres and using the two ML techniques, we rank the features (in a decreasing order) according to their impact (F-measure obtained) using FBVS (see 4.3). This results in a ranked list of features for each ML approach and data genre per class. Once the features are ranked, we incrementally experiment with the features in the order of the ranking, i.e. train with the first feature and measure the performance on the tuning data, then train with the second together with the first feature, i.e. the first two features and measure performance, then the first three features and so on.

Feats	PER	GPE	ORG	FAC	VEH/WEA
LEX_1	16	12	12	15	4
LEX_2	3	15	7	12	5
LEX_3	10	6	15	10	6
LEX_4	7	16	4	8	7
LEX_5	15	14	16	16	8
LEX_6	12	4	10	9	9
GAZ	14	7	9	11	3
BPC	4	13	13	6	1
POS	1	5	1	4	16
NAT	8	3	2	3	15
M_{ASP}	13	2	5	2	10
M_{PER}	11	11	3	5	14
M_{DEF}	9	9	6	7	11
M_{GEN}	5	8	11	13	12
M_{NUM}	6	10	14	14	13
CAP	2	1	8	1	2

Table 3: Ranked features according to FBVS using SVM for each NE class

4. Feature set/class generalization : Finally, we pick the first *n* features that yield the best converging performance (after which additional features do not impact performance or cause it to deteriorate). We use the top *n* features to tag the test data and compare the results against the system when it is trained on the whole feature set.

5.3 Individual Features Experiments

After running experiments using each feature individually, each result is considered an expert (the obtained F-measure is the weight in this framework).

Our goal is to find a general ranking of the features for each ML approach and each class. Table 3 shows the obtained rankings of the features for each class using SVM. It is worth noting that the obtained CRF rankings are very similar to those yielded by using SVM. We note that there are no specific features that have proven to be useless for all classes and ML approaches.

5.4 Feature set/class Experiments

We combine the features per NE class incrementally. Since the total number of features is 16, each ML classifier is trained and evaluated on the tuning data 16 times for each genre. A best number of features per class per genre per ML technique is determined based on the highest yielded $F_{\beta=1}$. Finally, the last step is combining the outputs of the different classifiers for all the classes. In case of conflicts, where the same token is tagged as two different NE classes, we use a simple heuristic based on the classifier precision for that specific tag, favoring the tag with the highest precision.

Table 4 illustrates the obtained results. For each data set and each genre it shows the F-measure obtained using the best feature set and ML approach. We show results for both the dev and test data using the optimal number of features **Best Feat-Set/ML** contrasted against the system when using all 16 features per class **All Feats/ML**. The table also illustrates three baseline results on the test data only. **FreqBaseline**: For this baseline, we assign a test token the most frequent tag observed for it in the training data, if a test token is not observed in the training data, it is assigned the most frequent tag which is the O tag. **MLBaseline**: In this baseline setting, we train an NER system with the full 16 features for all the NE classes at once. We use the two different ML approaches yielding two baselines: **MLBaseline**$_{SVM}$ and **MLBaseline**$_{CRF}$.
It is important to note the difference between the **All Feats/ML** setting and the **MLBaseline** setting. In the former, **All Feats/ML**, all 16 features are used per class in a 4-way classifier system and then the classifications are combined and the conflicts are resolved using our simple heuristic while in the latter case of **MLBaseline** the classes are trained together with all 16 features for all classes in one system. Since different feature-sets and different ML approaches are used and combined for each experiment, it is not possible to present the number of features used in each experiment in Table 4. However, Table 5 shows the number of features and the ML approach used for each genre and NE class.

6 Discussion and Error Analysis

As illustrated in Table 5, SVM outperformed CRF on most of the classes. Interestingly, CRF tends to model the ORG and FAC entities better than SVM. Hence, it is not possible to give a final word on the superiority of SVM or CRF in the NER task, and it is necessary to conduct a per class study, as the one we present in this paper, in order to determine the right ML approach and features to use for each class. Therefore, our best global NER system combined

		ACE 2003		*ACE 2004*			*ACE 2005*		
		BN	*NW*	*BN*	*NW*	*ATB*	*BN*	*NW*	*WL*
	FreqBaseline	73.74	67.61	62.17	51.67	62.94	70.18	57.17	27.66
	MLBaseline$_{SVM}$	80.58	76.37	74.21	71.11	73.14	79.3	73.9	54.68
	MLBaseline$_{CRF}$	81.02	76.18	74.67	71.8	73.04	80.13	74.75	55.32
dev	**Best Feat-set/ML**	**83.41**	**79.11**	**76.9**	**72.9**	**74.82**	**81.42**	**76.07**	**54.49**
dev	**All Feats. SVM**	81.79	77.99	75.49	71.8	73.71	80.87	75.69	53.73
dev	**All Feats. CRF**	81.76	76.6	76.26	71.85	74.19	79.66	74.83	36.11
test	**Best Feat-set/ML**	**83.5**	**78.9**	**76.7**	**72.4**	**73.5**	**81.31**	**75.3**	**57.3**
test	**All Feats. SVM**	81.76	77.27	74.71	71.16	73.63	81.1	72.41	55.58
test	**All Feats. CRF**	81.37	75.89	75.73	72.36	74.21	80.16	74.43	27.36

Table 4: Final Results obtained with selected features contrasted against all features combined

	BN		*NW*		*ATB*		*WL*	
	N	*ML*	*N*	*ML*	*N*	*ML*	*N*	*ML*
Person	12	SVM	14	SVM	9	SVM	11	SVM
Location	10	SVM	7	SVM	16	CRF	14	SVM
Organization	9	CRF	6	CRF	10	CRF	12	CRF
Facility	10	CRF	14	CRF	14	SVM	16	CRF
Vehicle	3	SVM	3	SVM	3	SVM	3	SVM
Weapon	3	SVM	3	SVM	3	SVM	3	SVM

Table 5: Number of features and ML approach used to obtain the best results

the results obtained from both ML approaches.

Table 4, shows that our **Best Feat-set/ML** setting outperforms the baselines and the **All Feats** {**SVM/CRF**} settings for all the data genres and sets forthe test data. Moreover, the **Best Feat-set/ML** setting outperforms both **All Feats** {**SVM/CRF**} settings for the dev data for all genres except for ACE2003 NW, where the difference is very small.

The results yielded from the ML baselines are comparable across all the data genres and the two ML approaches.

Comparing the global ML baseline systems against the All Feature Setting, we see that the **All Feats** setting consistently outperforms the **MLBaseline** settings except for ACE2005 NW data set. This suggests that training separate systems for the different NEs has some benefit over training in one global system.

Comparing the performance per genre across the different data sets. We note better performance across the board for BN data over NW per year. The worst results are yielded for ACE 2004 data for both BN and NW genres. There is no definitive conclusion that a specific ML approach is better suited for a specific data genre. We observe slightly better performance for the CRF ML approach in the **MLBaseline**$_{CRF}$ condition for both BN and NW.

The worst performance is yielded for the WL data. This may be attributed to the small amount of training data available for this genre. Moreover the quality of the performance of the different feature extraction tools such as AMIRA (for POS tagging and BPC) and MADA (for the morphological features) are optimized for NW data genres, thereby yielding suboptimal performance on the WL genre, leading to more noise than signal for training. However, comparing relative performance on this genre, we see a significant jump from the most frequent baseline **FreqBaseline** ($F_{\beta=1}$=27.66) to the best baseline **MLBaseline**$_{CRF}$ ($F_{\beta=1}$=55.32). We see a further significant improvement when the **Best Feat-set/ML** setting is applied yielding an $F_{\beta=1}$=57.3. Interestingly, however the **MLBaseline**$_{CRF}$ yields a much better performance ($F_{\beta=1}$=55.32) than **All Feats CRF** with an $F_{\beta=1}$=27.36. This may indicate that a global system that trains all classes at once using CRF for sparse data is better than training separate classifiers and then combining the outputs. It is worth noting the difference between **MLBaseline**$_{SVM}$ and **All Feats SVM**, F$_{\beta=1}$=54.68 and F$_{\beta=1}$=55.58, respectively. This result suggests that SVM are more robust to less training data as illustrated in the case of the individual classifiers in the latter setting.

Comparing dev and test performance, we note that the overall results on the dev data are better than those obtained on the test data, which is expected given that the weights for the FBVS ranking are derived based on the dev data used as a tuning set. The only counter example for this trend is with the WL data genre, where the test data yields a significantly higher performance for all the conditions except for **All Feats CRF**.

As observed in Table 3, the ranking of the individual features could be very different for two NE classes. For instance, the BPC is ranked 4th for the PER class and is ranked 13th for GPE and ORG classes. The disparity in ranking for the same individual features strongly suggests that using the same features for all the classes cannot lead to a global optimal classifier. With regards to morphological features, we note in Table 3, that Definiteness, M_{DEF}, is helpful for all the NE classification systems, by virtue of being included for all optimal systems for all NE classification systems. Aspect, M_{ASP}, is useful for all classes except PER. Moreover, M_{GEN} and M_{NUM}, corresponding to Gender and Number, respectively, contributed significantly to the increase in recall for PER and GPE classes. Finally, the Person feature, M_{PER} contributed mostly to improving the classification of ORG and FAC classes. Accordingly, observing these results, contrary to previous results by (Farber et al., 2004), our results strongly suggest the significant impact morphological features have on Arabic NER, if applied at the right level of granularity.

Inconsistencies in the data lead to many of the observed errors. The problem is that the ACE data is annotated primarily for a mention detection task which leads to the same exact words not being annotated consistently. For instance, the word 'Palestinians' would sometimes be annotated as a GPE class while in similar other contexts it is not annotated as a named entity at all. Since we did not manually correct these cases, the classifiers are left with mixed signals. The VEH and WEA classes both exhibit a uniform ranking for all the features and yield a very low performance. This is mainly attributed to the fact that they appear very rarely in the training data. For instance, in the ACE 2005, BN genre, there are 1707 instances of the class PER, 1777 of GPE, 103 of ORG, 106 of FAC and only 4 for WEA and 24 for VEH.

7 Conclusions and Future Directions

We described the performance yielded using language-dependent and language independent features in SVM and CRF for the NER task on different standard Arabic data-sets comprising different genres. We have measured the impact of each feature individually on each class, we ranked them according to their impact using the Fuzzy Borda Voting Scheme, and then performed an incremental features' selection considering each time the N best features.

We reported the importance of each feature for each class and then the performance obtained when the best feature-set is used. Our experiments yield state of the art performance significantly outperforming the baseline. Our best results achieve an F$_{\beta=1}$ score of 83.5 for the ACE 2003 BN data. Our ACE2005 results are state of the art when compared to the best system to date. It is worth noting that these obtained results are trained on less data since we train only on 3 folds vs the standard 4 folds. Our results show that the SVM and CRF have very similar behaviors. However, SVM showed more robust performance in our system using data with very random contexts, namely for the WL data, i.e. Weblogs. We definitively illustrate that correctly exploiting morphological features for languages with rich morphological structures yields state of the art performance. For future work, we intend to investigate the use of automatic feature selection methods on the same data.

Acknowledgments The authors would like to thank the reviewers for their detailed constructive comments. We would like to thank MCyT TIN2006-15265-C06-04 and PCI-AECI A/010317/07 research projects for partially funding this work. Mona Diab would like to acknowledge DARPA GALE Grant Contract No. HR0011-06-C-0023 for partially funding this work.

References

Bogdan Babych and Anthony Hartley. 2003. *Improving Machine Translation Quality with Automatic Named Entity Recognition*. In *Proc. of EACL-EAMT*. Budapest.

Yassine Benajiba and Paolo Rosso. 2008. *Arabic Named Entity Recognition using Conditional Random Fields*. In *Proc. of Workshop on HLT & NLP within the Arabic World, LREC'08*.

Yassine Benajiba, Paolo Rosso and José Miguel Benedí. 2007. *ANERsys: An Arabic Named Entity Recognition system based on Maximum Entropy*. In *Proc. of CICLing-2007*, Springer-Verlag, LNCS(4394), pp. 143-153.

Yassine Benajiba and Paolo Rosso. 2007. *ANERsys 2.0 : Conquering the NER task for the Arabic language by combining the Maximum Entropy with POS-tag information*. In *Proc. of Workshop on Natural Language-Independent Engineering, IICAI-2007*.

Oliver Bender, Franz Josef Och, and Hermann Ney. 2003. *Maximum Entropy Models For Named Entity Recognition*. In *Proc. of CoNLL-2003*. Edmonton, Canada.

Tim Buckwalter. 2002. *Buckwalter Arabic Morphological Analyzer*. In *Linguistic Data Consortium. (LDC2002L49)*.

Mona Diab, Kadri Hacioglu and Daniel Jurafsky. 2007. *Arabic Computational Morphology: Knowledge-based and Empirical Methods*, chapter 9, pp. 159–179. Abdelhadi Soudi, Antal van den Bosch and Gunter Neumann (Eds.), Springer.

Benjamin Farber, Dayne Freitag, Nizar Habash and Owen Rambow. 2008. *Improving NER in Arabic Using a Morphological Tagger.* In *Proc. of LREC'08*.

Sergio Ferrández, Óscar Ferrández, Antonio Ferrández and Rafael Muñoz. 2007. *The Importance of Named Entities in Cross-Lingual Question Answering* In *Proc. of RANLP'07*.

José Luis García Lapresta and Miguel Martínez Panero 2002. *Borda Count Versus Approval Voting: A Fuzzy Approach*. Public Choice, 112(1-2):pp. 167–184.

Nizar Habash and Owen Rambow. 2005. *Arabic Tokenization, Part-Of-Speech Tagging and Morphological Disambiguation in One Fell Swoop*. In *Proc. of Workshop of Computational Approaches to Semitic Languages, ACL-2005*.

John Lafferty, Andrew McCallum and Fernando Pereira. 2001. *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data*. In *Proc. of ICML-2001*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. *Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition*. In *Proc. of CoNLL-2003*. pp. 142–147.

Hiroyuki Toda and Ryoji Kataoka. 2005. *A Search Result Clustering Method using Informatively Named Entities.*. In *Proc. of the 7th ACM International Workshop on Web Information and Data Management*.

Q. Tri Tran, T.X. Thao Pham, Q. Hung Ngo,Dien Dinh, and Nigel Collier. 2007. *Named Entity Recognition in Vietnamese documents*. *Progress in Informatics Journal*. 2007.

Vladimir Vapnik. 1995. *The Nature of Statistical Learning Theory. Springer Verlag*.

Understanding the Value of Features for Coreference Resolution

Eric Bengtson **Dan Roth**
Department of Computer Science
University of Illinois
Urbana, IL 61801
{ebengt2,danr}@illinois.edu

Abstract

In recent years there has been substantial work on the important problem of coreference resolution, most of which has concentrated on the development of new models and algorithmic techniques. These works often show that complex models improve over a weak pairwise baseline. However, less attention has been given to the importance of selecting strong features to support learning a coreference model.

This paper describes a rather simple pairwise classification model for coreference resolution, developed with a well-designed set of features. We show that this produces a state-of-the-art system that outperforms systems built with complex models. We suggest that our system can be used as a baseline for the development of more complex models – which may have less impact when a more robust set of features is used. The paper also presents an ablation study and discusses the relative contributions of various features.

1 Introduction

Coreference resolution is the task of grouping all the mentions of entities[1] in a document into equivalence classes so that all the mentions in a given class refer to the same discourse entity. For example, given the sentence (where the head noun of each mention is subscripted)

> *An American$_1$ official$_2$ announced that American$_1$ President$_3$ Bill Clinton$_3$ met his$_3$ Russian$_4$ counterpart$_5$, Vladimir Putin$_5$, today.*

the task is to group the mentions so that those referring to the same entity are placed together into an equivalence class.

Many NLP tasks detect attributes, actions, and relations between discourse entities. In order to discover all information about a given entity, textual mentions of that entity must be grouped together. Thus coreference is an important prerequisite to such tasks as textual entailment and information extraction, among others.

Although coreference resolution has received much attention, that attention has not focused on the relative impact of high-quality features. Thus, while many structural innovations in the modeling approach have been made, those innovations have generally been tested on systems with features whose strength has not been established, and compared to weak pairwise baselines. As a result, it is possible that some modeling innovations may have less impact or applicability when applied to a stronger baseline system.

This paper introduces a rather simple but state-of-the-art system, which we intend to be used as a strong baseline to evaluate the impact of structural innovations. To this end, we combine an effective coreference classification model with a strong set of features, and present an ablation study to show the relative impact of a variety of features.

As we show, this combination of a pairwise model and strong features produces a 1.5 percent-

[1]We follow the ACE (NIST, 2004) terminology: A noun phrase referring to a discourse entity is called a *mention*, and an equivalence class is called an *entity*.

age point increase in B-Cubed F-Score over a complex model in the state-of-the-art system by Culotta et al. (2007), although their system uses a complex, non-pairwise model, computing features over partial clusters of mentions.

2 A Pairwise Coreference Model

Given a document and a set of mentions, coreference resolution is the task of grouping the mentions into equivalence classes, so that each equivalence class contains exactly those mentions that refer to the same discourse entity. The number of equivalence classes is not specified in advance, but is bounded by the number of mentions.

In this paper, we view coreference resolution as a graph problem: Given a set of mentions and their context as nodes, generate a set of edges such that any two mentions that belong in the same equivalence class are connected by some path in the graph. We construct this entity-mention graph by learning to decide for each mention which preceding mention, if any, belongs in the same equivalence class; this approach is commonly called the pairwise coreference model (Soon et al., 2001). To decide whether two mentions should be linked in the graph, we learn a pairwise coreference function *pc* that produces a value indicating the probability that the two mentions should be placed in the same equivalence class.

The remainder of this section first discusses how this function is used as part of a document-level coreference decision model and then describes how we learn the *pc* function.

2.1 Document-Level Decision Model

Given a document d and a pairwise coreference scoring function pc that maps an ordered pair of mentions to a value indicating the probability that they are coreferential (see Section 2.2), we generate a coreference graph G_d according to the Best-Link decision model (Ng and Cardie, 2002b) as follows:

For each mention m in document d, let B_m be the set of mentions appearing before m in d. Let a be the highest scoring antecedent:

$$a = \underset{b \in B_m}{\operatorname{argmax}}(pc(b, m)).$$

If $pc(a, m)$ is above a threshold chosen as described in Section 4.4, we add the edge (a, m) to the coreference graph G_d.

The resulting graph contains connected components, each representing one equivalence class, with all the mentions in the component referring to the same entity. This technique permits us to learn to detect some links between mentions while being agnostic about whether other mentions are linked, and yet via the transitive closure of all links we can still determine the equivalence classes.

We also require that no non-pronoun can refer back to a pronoun: If m is not a pronoun, we do not consider pronouns as candidate antecedents.

2.1.1 Related Models

For pairwise models, it is common to choose the best antecedent for a given mention (thereby imposing the constraint that each mention has at most one antecedent); however, the method of deciding which is the best antecedent varies.

Soon et al. (2001) use the Closest-Link method: They select as an antecedent the closest preceding mention that is predicted coreferential by a pairwise coreference module; this is equivalent to choosing the closest mention whose pc value is above a threshold. Best-Link was shown to outperform Closest-Link in an experiment by Ng and Cardie (2002b). Our model differs from that of Ng and Cardie in that we impose the constraint that non-pronouns cannot refer back to pronouns, and in that we use as training examples all ordered pairs of mentions, subject to the constraint above.

Culotta et al. (2007) introduced a model that predicts whether a pair of equivalence classes should be merged, using features computed over all the mentions in both classes. Since the number of possible classes is exponential in the number of mentions, they use heuristics to select training examples. Our method does not require determining which equivalence classes should be considered as examples.

2.2 Pairwise Coreference Function

Learning the pairwise scoring function pc is a crucial issue for the pairwise coreference model. We apply machine learning techniques to learn from examples a function pc that takes as input an ordered pair of mentions (a, m) such that a precedes m in the document, and produces as output a value that is interpreted as the conditional probability that m and a belong in the same equivalence class.

2.2.1 Training Example Selection

The ACE training data provides the equivalence classes for mentions. However, for some pairs of mentions from an equivalence class, there is little or no direct evidence in the text that the mentions are coreferential. Therefore, training *pc* on all pairs of mentions within an equivalence class may not lead to a good predictor. Thus, for each mention m we select from m's equivalence class the closest preceding mention a and present the pair (a, m) as a positive training example, under the assumption that there is more direct evidence in the text for the existence of this edge than for other edges. This is similar to the technique of Ng and Cardie (2002b). For each m, we generate negative examples (a, m) for all mentions a that precede m and are not in the same equivalence class. Note that in doing so we generate more negative examples than positive ones.

Since we never apply *pc* to a pair where the first mention is a pronoun and the second is not a pronoun, we do not train on examples of this form.

2.2.2 Learning Pairwise Coreference Scoring Model

We learn the pairwise coreference function using an averaged perceptron learning algorithm (Freund and Schapire, 1998) – we use the regularized version in Learning Based Java[2] (Rizzolo and Roth, 2007).

3 Features

The performance of the document-level coreference model depends on the quality of the pairwise coreference function *pc*. Beyond the training paradigm described earlier, the quality of *pc* depends on the features used.

We divide the features into categories, based on their function. A full list of features and their categories is given in Table 2. In addition to these boolean features, we also use the conjunctions of all pairs of features.[3]

[2]LBJ code is available at `http://L2R.cs.uiuc.edu/~cogcomp/asoftware.php?skey=LBJ`

[3]The package of all features used is available at `http://L2R.cs.uiuc.edu/~cogcomp/asoftware.php?skey=LBJ#features`.

In the following description, the term *head* means the head noun phrase of a mention; the *extent* is the largest noun phrase headed by the head noun phrase.

3.1 Mention Types

The type of a mention indicates whether it is a proper noun, a common noun, or a pronoun. This feature, when conjoined with others, allows us to give different weight to a feature depending on whether it is being applied to a proper name or a pronoun. For our experiments in Section 5, we use gold mention types as is done by Culotta et al. (2007) and Luo and Zitouni (2005).

Note that in the experiments described in Section 6 we predict the mention types as described there and do not use any gold data. The mention type feature is used in all experiments.

3.2 String Relation Features

String relation features indicate whether two strings share some property, such as one being the substring of another or both sharing a modifier word. Features are listed in Table 1. Modifiers are limited to those occurring before the head.

Feature	Definition
Head match	$head_i == head_j$
Extent match	$extent_i == extent_j$
Substring	$head_i$ substring of $head_j$
Modifiers Match	$mod_i ==$ ($head_j$ or mod_j)
Alias	$acronym(head_i) == head_j$ or $lastname_i == lastname_j$

Table 1: String Relation Features

3.3 Semantic Features

Another class of features captures the semantic relation between two words. Specifically, we check whether gender or number match, or whether the mentions are synonyms, antonyms, or hypernyms. We also check the relationship of modifiers that share a hypernym. Descriptions of the methods for computing these features are described next.

Gender Match We determine the gender (male, female, or neuter) of the two phrases, and report whether they match (true, false, or unknown). For

Category	Feature	Source
Mention Types	Mention Type Pair	Annotation and tokens
String Relations	Head Match	Tokens
	Extent Match	Tokens
	Substring	Tokens
	Modifiers Match	Tokens
	Alias	Tokens and lists
Semantic	Gender Match	WordNet and lists
	Number Match	WordNet and lists
	Synonyms	WordNet
	Antonyms	WordNet
	Hypernyms	WordNet
	Both Speak	Context
Relative Location	Apposition	Positions and context
	Relative Pronoun	Positions and tokens
	Distances	Positions
Learned	Anaphoricity	Learned
	Name Modifiers Predicted Match	Learned
Aligned Modifiers	Aligned Modifiers Relation	WordNet and lists
Memorization	Last Words	Tokens
Predicted Entity Types	Entity Types Match	Annotation and tokens
	Entity Type Pair	WordNet and tokens

Table 2: Features by Category

a proper name, gender is determined by the existence of *mr*, *ms*, *mrs*, or the gender of the first name. If only a last name is found, the phrase is considered to refer to a person. If the name is found in a comprehensive list of cities or countries, or ends with an organization ending such as *inc*, then the gender is neuter. In the case of a common noun phrase, the phrase is looked up in WordNet (Fellbaum, 1998), and it is assigned a gender according to whether *male*, *female*, *person*, *artifact*, *location*, or *group* (the last three correspond to neuter) is found in the hypernym tree. The gender of a pronoun is looked up in a table.

Number Match Number is determined as follows: Phrases starting with the words *a*, *an*, or *this* are singular; *those*, *these*, or *some* indicate plural. Names not containing *and* are singular. Common nouns are checked against extensive lists of singular and plural nouns – words found in neither or both lists have unknown number. Finally, if the number is unknown yet the two mentions have the same spelling, they are assumed to have the same number.

WordNet Features We check whether any sense of one head noun phrase is a synonym, antonym, or hypernym of any sense of the other. We also check whether any sense of the phrases share a hypernym, after dropping *entity*, *abstraction*, *physical entity*, *object*, *whole*, *artifact*, and *group* from the senses, since they are close to the root of the hypernym tree.

Modifiers Match Determines whether the text before the head of a mention matches the head or the text before the head of the other mention.

Both Mentions Speak True if both mentions appear within two words of a verb meaning *to say*. Being in a window of size two is an approximation to being a syntactic subject of such a verb. This feature is a proxy for having similar semantic types.

3.4 Relative Location Features

Additional evidence is derived from the relative location of the two mentions. We thus measure distance (quantized as multiple boolean features of the form [$distance \geq i$]) for all i up to the distance and less than some maximum, using units of compatible mentions, and whether the mentions are in the same sentence. We also detect apposition (mentions separated by a comma). For details, see Table 3.

Feature	Definition
Distance	In same sentence # compatible mentions
Apposition	m_1, m_2 found
Relative Pronoun	Apposition and m_2 is PRO

Table 3: Location Features. Compatible mentions are those having the same gender and number.

3.5 Learned Features

Modifier Names If the mentions are both modified by other proper names, use a basic coreference classifier to determine whether the modifiers are coreferential. This basic classifier is trained using Mention Types, String Relations, Semantic Features, Apposition, Relative Pronoun, and Both Speak. For each mention m, examples are generated with the closest antecedent a to form a positive example, and every mention between a and m to form negative examples.

Anaphoricity Ng and Cardie (2002a) and Denis and Baldridge (2007) show that when used effectively, explicitly predicting anaphoricity can be helpful. Thus, we learn a separate classifier to detect whether a mention is anaphoric (that is, whether it is not the first mention in its equivalence class), and use that classifier's output as a feature for the coreference model. Features for the anaphoricity classifier include the mention type, whether the mention appears in a quotation, the text of the first word of the extent, the text of the first word after the head (if that word is part of the extent), whether there is a longer mention preceding this mention and having the same head text, whether any preceding mention has the same extent text, and whether any preceding mention has the same text from beginning of the extent to end of the head. Conjunctions of all pairs of these features are also used. This classifier predicts anaphoricity with about 82% accuracy.

3.6 Aligned Modifiers

We determine the relationship of any pair of modifiers that share a hypernym. Each aligned pair may have one of the following relations: match, substring, synonyms, hypernyms, antonyms, or mismatch. Mismatch is defined as none of the above. We restrict modifiers to single nouns and adjectives occurring before the head noun phrase.

3.7 Memorization Features

We allow our system to learn which pairs of nouns tend to be used to mention the same entity. For example, *President* and *he* often refer to *Bush* but *she* and *Prime Minister* rarely do, if ever. To enable the system to learn such patterns, we treat the presence or absence of each pair of final head nouns, one from each mention of an example, as a feature.

3.8 Predicted Entity Type

We predict the entity type (*person*, *organization*, *geo-political entity*, *location*, *facility*, *weapon*, or *vehicle*) as follows: If a proper name, we check a list of personal first names, and a short list of honorary titles (e.g. *mr*) to determine if the mention is a person. Otherwise we look in lists of personal last names drawn from US census data, and in lists of cities, states, countries, organizations, corporations, sports teams, universities, political parties, and organization endings (e.g. *inc* or *corp*). If found in exactly one list, we return the appropriate type. We return *unknown* if found in multiple lists because the lists are quite comprehensive and may have significant overlap.

For common nouns, we look at the hypernym tree for one of the following: *person*, *political unit*, *location*, *organization*, *weapon*, *vehicle*, *industrial plant*, and *facility*. If any is found, we return the appropriate type. If multiple are found, we sort as in the above list.

For personal pronouns, we recognize the entity as a person; otherwise we specify unknown.

This computation is used as part of the following two features.

Entity Type Match This feature checks to see whether the predicted entity types match. The result is true if the types are identical, false if they are different, and unknown if at least one type is unknown.

Entity Type Conjunctions This feature indicates the presence of the pair of predicted entity types for the two mentions, except that if either word is a pronoun, the word token replaces the type in the pair. Since we do this replacement for entity types, we also add a similar feature for mention types here. These features are boolean: For any given pair, a feature is active if that pair describes the example.

3.9 Related Work

Many of our features are similar to those described in Culotta et al. (2007). This includes Mention Types, String Relation Features, Gender and Number Match, WordNet Features, Alias, Apposition, Relative Pronoun, and Both Mentions Speak. The implementations of those features may vary from those of other systems. Anaphoricity has been proposed as a part of the model in several systems, including Ng and Cardie (2002a), but we are not aware of it being used as a feature for a learning algorithm. Distances have been used in e.g. Luo et al. (2004). However, we are not aware of any system using the number of compatible mentions as a distance.

4 Experimental Study

4.1 Corpus

We use the official ACE 2004 English training data (NIST, 2004). Much work has been done on coreference in several languages, but for this work we focus on English text. We split the corpus into three sets: Train, Dev, and Test. Our test set contains the same 107 documents as Culotta et al. (2007). Our training set is a random 80% of the 336 documents in their training set and our Dev set is the remaining 20%.

For our ablation study, we further randomly split our development set into two evenly sized parts, Dev-Tune and Dev-Eval. For each experiment, we set the parameters of our algorithm to optimize B-Cubed F-Score using Dev-Tune, and use those parameters to evaluate on the Dev-Eval data.

4.2 Preprocessing

For the experiments in Section 5, following Culotta et al. (2007), to make experiments more comparable across systems, we assume that perfect mention boundaries and mention type labels are given. We do not use any other gold annotated input at evaluation time. In Section 6 experiments we do not use any gold annotated input and do not assume mention types or boundaries are given. In all experiments we automatically split words and sentences using our preprocessing tools.[4]

4.3 Evaluation Scores

B-Cubed F-Score We evaluate over the commonly used B-Cubed F-Score (Bagga and Baldwin, 1998), which is a measure of the overlap of predicted clusters and true clusters. It is computed as the harmonic mean of precision (P),

$$P = \frac{1}{N} \sum_{d \in D} \left(\sum_{m \in d} \left(\frac{c_m}{p_m} \right) \right), \qquad (1)$$

and recall (R),

$$R = \frac{1}{N} \sum_{d \in D} \left(\sum_{m \in d} \left(\frac{c_m}{t_m} \right) \right), \qquad (2)$$

where c_m is the number of mentions appearing both in m's predicted cluster and in m's true cluster, p_m is the size of the predicted cluster containing m, and t_m is the size of m's true cluster. Finally, d represents a document from the set D, and N is the total number of mentions in D.

B-Cubed F-Score has the advantage of being able to measure the impact of singleton entities, and of giving more weight to the splitting or merging of larger entities. It also gives equal weight to all types of entities and mentions. For these reasons, we report our results using B-Cubed F-Score.

MUC F-Score We also provide results using the official MUC scoring algorithm (Vilain et al., 1995). The MUC F-score is also the harmonic mean of precision and recall. However, the MUC precision counts precision errors by computing the minimum number of links that must be added to ensure that all mentions referring to a given entity are connected in the graph. Recall errors are the number of links that must be removed to ensure that no two mentions referring to different entities are connected in the graph.

[4]The code is available at `http://L2R.cs.uiuc.edu/~cogcomp/tools.php`

4.4 Learning Algorithm Details

We train a regularized average perceptron using examples selected as described in Section 2.2.1. The learning rate is 0.1 and the regularization parameter (separator thickness) is 3.5. At training time, we use a threshold of 0.0, but when evaluating, we select parameters to optimize B-Cubed F-Score on a held-out development set. We sample all even integer thresholds from -16 to 8. We choose the number of rounds of training similarly, allowing any number from one to twenty.

5 Results

	Precision	Recall	B^3 F
Culotta et al.	86.7	73.2	79.3
Current Work	88.3	74.5	**80.8**

Table 4: Evaluation on unseen Test Data using B^3 score. Shows that our system outperforms the advanced system of Culotta et al. The improvement is statistically significant at the $p = 0.05$ level according to a non-parametric bootstrapping percentile test.

In Table 4, we compare our performance against a system that is comparable to ours: Both use gold mention boundaries and types, evaluate using B-Cubed F-Score, and have the same training and test data split. Culotta et al. (2007) is the best comparable system of which we are aware.

Our results show that a pairwise model with strong features outperforms a state-of-the-art system with a more complex model.

MUC Score We evaluate the performance of our system using the official MUC score in Table 5.

MUC Precision	MUC Recall	MUC F
82.7	69.9	75.8

Table 5: Evaluation of our system on unseen Test Data using MUC score.

5.1 Analysis of Feature Contributions

In Table 6 we show the relative impact of various features. We report data on Dev-Eval, to avoid the possibility of overfitting by feature selection. The parameters of the algorithm are chosen to maximize the BCubed F-Score on the Dev-Tune data. Note that since we report results on Dev-Eval, the results in Table 6 are not directly comparable with Culotta et al. (2007). For comparable results, see Table 4 and the discussion above.

Our ablation study shows the impact of various classes of features, indicating that almost all the features help, although some more than others. It also illustrates that some features contribute more to precision, others more to recall. For example, aligned modifiers contribute primarily to precision, whereas our learned features and our apposition features contribute to recall. This information can be useful when designing a coreference system in an application where recall is more important than precision, or vice versa.

We examine the effect of some important features, selecting those that provide a substantial improvement in precision, recall, or both. For each such feature we examine the rate of coreference amongst mention pairs for which the feature is active, compared with the overall rate of coreference. We also show examples on which the coreference systems differ depending on the presence or absence of a feature.

Apposition This feature checks whether two mentions are separated by only a comma, and it increases B-Cubed F-Score by about one percentage point. We hypothesize that proper names and common noun phrases link primarily through apposition, and that apposition is thus a significant feature for good coreference resolution.

When this feature is active 36% of the examples are coreferential, whereas only 6% of all examples are coreferential. Looking at some examples our system begins to get right when apposition is added, we find the phrase

> *Israel's Deputy Defense Minister, Ephraim Sneh.*

Upon adding apposition, our system begins to correctly associate *Israel's Deputy Defense Minister* with *Ephraim Sneh*. Likewise in the phrase

> *The court president, Ronald Sutherland*,

the system correctly associates *The court president* with *Ronald Sutherland* when they appear in an appositive relation in the text. In addition, our system

	Precision	Recall	B-Cubed F
String Similarity	86.88	67.17	75.76
+ Semantic Features	85.34	69.30	**76.49**
+ Apposition	89.77	67.53	77.07
+ Relative Pronoun	88.76	68.97	77.62
+ Distances	89.62	71.93	79.81
+ Learned Features	87.37	74.51	**80.43**
+ Aligned Modifiers	88.70	74.30	80.86
+ Memorization	86.57	75.59	80.71
+ Predicted Entity Types	87.92	76.46	81.79

Table 6: Contribution of Features as evaluated on a development set. Bold results are significantly better than the previous line at the $p = 0.05$ level according to a paired non-parametric bootstrapping percentile test. These results show the importance of Distance, Entity Type, and Apposition features.

begins correctly associating relative pronouns such as *who* with their referents in phrases like

Sheikh Abbad, who died 500 years ago.

although an explicit relative pronoun feature is added only later.

Although this feature may lead the system to link comma separated lists of entities due to misinterpretation of the comma, for example *Wyoming* and *western South Dakota* in a list of locations, we believe this can be avoided by refining the apposition feature to ignore lists.

Relative Pronoun Next we investigate the relative pronoun feature. With this feature active, 93% of examples were positive, indicating the precision of this feature. Looking to examples, we find *who* in

the official, who wished to remain anonymous

is properly linked, as is *that* in

nuclear warheads that can be fitted to missiles.

Distances Our distance features measure separation of two mentions in number of compatible mentions (quantized), and whether the mentions are in the same sentence. Distance features are important for a system that makes links based on the best pairwise coreference value rather than implicitly incorporating distance by linking only the closest pair whose score is above a threshold, as done by e.g. Soon et al. (2001).

Looking at examples, we find that adding distances allows the system to associate the pronoun *it* with *this missile* not separated by any mentions, rather than *Tehran*, which is separated from *it* by many mentions.

Predicted Entity Types Since no two mentions can have different entity types (person, organization, geo-political entity, etc.) and be coreferential, this feature has strong discriminative power. When the entity types match, 13% of examples are positive compared to only 6% of examples in general. Qualitatively, the entity type prediction correctly recognizes *the Gulf region* as a geo-political entity, and *He* as a person, and thus prevents linking the two. Likewise, the system discerns *Baghdad* from *ambassador* due to the entity type. However, in some cases an identity type match can cause the system to be overly confident in a bad match, as in the case of a *palestinian state* identified with *holy Jerusalem* on the basis of proximity and shared entity type. This type of example may require some additional world knowledge or deeper comprehension of the document.

6 End-to-End Coreference

The ultimate goal for a coreference system is to process unannotated text. We use the term *end-to-end coreference* for a system capable of determining coreference on plain text. We describe the challenges associated with an end-to-end system, describe our approach, and report results below.

6.1 Challenges

Developing an end-to-end system requires detecting and classifying mentions, which may degrade coreference results. One challenge in detecting mentions is that they are often heavily nested. Additionally, there are issues with evaluating an end-to-end system against a gold standard corpus, resulting from the possibility of mismatches in mention boundaries, missing mentions, and additional mentions detected, along with the need to align detected mentions to their counterparts in the annotated data.

6.2 Approach

We resolve coreference on unannotated text as follows: First we detect mention heads following a state of the art chunking approach (Punyakanok and Roth, 2001) using standard features. This results in a 90% F_1 head detector. Next, we detect the extent boundaries for each head using a learned classifier. This is followed by determining whether a mention is a proper name, common noun phrase, prenominal modifier, or pronoun using a learned mention type classifier that. Finally, we apply our coreference algorithm described above.

6.3 Evaluation and Results

To evaluate, we align the heads of the detected mentions to the gold standard heads greedily based on number of overlapping words. We choose not to impute errors to the coreference system for mentions that were not detected or for spuriously detected mentions (following Ji et al. (2005) and others). Although this evaluation is lenient, given that the mention detection component performs at over 90% F_1, we believe it provides a realistic measure for the performance of the end-to-end system and focuses the evaluation on the coreference component. The results of our end-to-end coreference system are shown in Table 7.

	Precision	Recall	B^3 F
End-to-End System	84.91	72.53	78.24

Table 7: Coreference results using detected mentions on unseen Test Data.

7 Conclusion

We described and evaluated a state-of-the-art coreference system based on a pairwise model and strong features. While previous work showed the impact of complex models on a weak pairwise baseline, the applicability and impact of such models on a strong baseline system such as ours remains uncertain. We also studied and demonstrated the relative value of various types of features, showing in particular the importance of distance and apposition features, and showing which features impact precision or recall more. Finally, we showed an end-to-end system capable of determining coreference in a plain text document.

Acknowledgments

We would like to thank Ming-Wei Chang, Michael Connor, Alexandre Klementiev, Nick Rizzolo, Kevin Small, and the anonymous reviewers for their insightful comments. This work is partly supported by NSF grant SoD-HCER-0613885 and a grant from Boeing.

References

A. Bagga and B. Baldwin. 1998. Algorithms for scoring coreference chains. In *MUC7*.

A. Culotta, M. Wick, R. Hall, and A. McCallum. 2007. First-order probabilistic models for coreference resolution. In *HLT/NAACL*, pages 81–88.

P. Denis and J. Baldridge. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. In *HLT/NAACL*, pages 236–243, Rochester, New York.

C. Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Y. Freund and R. E. Schapire. 1998. Large margin classification using the Perceptron algorithm. In *COLT*, pages 209–217.

H. Ji, D. Westbrook, and R. Grishman. 2005. Using semantic relations to refine coreference decisions. In *EMNLP/HLT*, pages 17–24, Vancouver, British Columbia, Canada.

X. Luo and I. Zitouni. 2005. Multi-lingual coreference resolution with syntactic features. In *HLT/EMNLP*, pages 660–667, Vancouver, British Columbia, Canada.

X. Luo, A. Ittycheriah, H. Jing, N. Kambhatla, and S. Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the bell tree. In *ACL*, page 135, Morristown, NJ, USA.

V. Ng and C. Cardie. 2002a. Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. In *COLING-2002*.

V. Ng and C. Cardie. 2002b. Improving machine learning approaches to coreference resolution. In *ACL*.

NIST. 2004. The ace evaluation plan. www.nist.gov/speech/tests/ace/index.htm.

V. Punyakanok and D. Roth. 2001. The use of classifiers in sequential inference. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 995–1001. MIT Press.

N. Rizzolo and D. Roth. 2007. Modeling Discriminative Global Inference. In *Proceedings of the First International Conference on Semantic Computing (ICSC)*, pages 597–604, Irvine, California.

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

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

Selecting Sentences for Answering Complex Questions

Yllias Chali
University of Lethbridge
4401 University Drive
Lethbridge, Alberta, Canada, T1K 3M4
chali@cs.uleth.ca

Shafiq R. Joty
University of British Columbia
2366 Main Mall
Vancouver, B.C. Canada V6T 1Z4
rjoty@cs.ubc.ca

Abstract

Complex questions that require inferencing and synthesizing information from multiple documents can be seen as a kind of topic-oriented, informative multi-document summarization. In this paper, we have experimented with one empirical and two unsupervised statistical machine learning techniques: k-means and Expectation Maximization (EM), for computing relative importance of the sentences. However, the performance of these approaches depends entirely on the feature set used and the weighting of these features. We extracted different kinds of features (i.e. lexical, lexical semantic, cosine similarity, basic element, tree kernel based syntactic and shallow-semantic) for each of the document sentences in order to measure its importance and relevancy to the user query. We used a local search technique to learn the weights of the features. For all our methods of generating summaries, we have shown the effects of syntactic and shallow-semantic features over the bag of words (BOW) features.

1 Introduction

After having made substantial headway in factoid and list questions, researchers have turned their attention to more complex information needs that cannot be answered by simply extracting named entities (persons, organizations, locations, dates, etc.) from documents. For example, the question: *"Describe the after-effects of cyclone Sidr-Nov 2007 in Bangladesh"* requires inferencing and synthesizing information from multiple documents. This information synthesis in NLP can be seen as a kind of topic-oriented, informative multi-document summarization, where the goal is to produce a single text as a compressed version of a set of documents with a minimum loss of relevant information.

In this paper, we experimented with one empirical and two well-known unsupervised statistical machine learning techniques: k-means and EM and evaluated their performance in generating topic-oriented summaries. However, the performance of these approaches depends entirely on the feature set used and the weighting of these features. We extracted different kinds of features (i.e. lexical, lexical semantic, cosine similarity, basic element, tree kernel based syntactic and shallow-semantic) for each of the document sentences in order to measure its importance and relevancy to the user query. We have used a gradient descent local search technique to learn the weights of the features. Traditionally, information extraction techniques are based on the BOW approach augmented by language modeling. But when the task requires the use of more complex semantics, the approaches based on only BOW are often inadequate to perform fine-level textual analysis. Some improvements on BOW are given by the use of dependency trees and syntactic parse trees (Hirao et al., 2004), (Punyakanok et al., 2004), (Zhang and Lee, 2003), but these, too are not adequate when dealing with complex questions whose answers are expressed by long and articulated sentences or even paragraphs. Shallow semantic representations, bearing a more compact information, could prevent the sparseness of deep structural approaches and the weakness of BOW models (Moschitti et al., 2007). Attempting an application of syntactic and semantic information to complex QA hence seems natural, as pinpointing the answer to a question relies on a deep understanding of the semantics of both. In more complex tasks such as computing the relatedness between the query sentences and the document sentences in order to generate query-focused summaries (or answers to complex questions), to our knowledge no study uses tree kernel functions to encode syntactic/semantic information. For all our methods of generating summaries (i.e. empirical, k-means and EM), we have shown the effects of syntactic and shallow-semantic features over the BOW features.

This paper is organized as follows: Section 2 focuses on the related work, Section 3 describes how the features are extracted, Section 4 discusses the scoring approaches, Section 5 discusses how we remove the redundant sentences before adding them to the summary, Section 6 describes our experimental study. We conclude and give future directions in Section 7.

2 Related Work

Researchers all over the world working on query-based summarization are trying different directions to see which methods provide the best results. The *LexRank* method addressed in (Erkan and Radev, 2004) was very successful in generic multi-document summarization. A topic-sensitive LexRank is proposed in (Otterbacher et al., 2005). As in LexRank, the set of sentences in a document cluster is represented as a graph, where nodes are sentences and links between the nodes are induced by a similarity relation between the sentences. Then the system ranked the sentences according to a random walk model defined in terms of both the inter-sentence similarities and the similarities of the sentences to the topic description or question.

The summarization methods based on lexical chain first extract the nouns, compound nouns and named entities as candidate words (Li et al., 2007). Then using WordNet, the systems find the semantic similarity between the nouns and compound nouns. After that, lexical chains are built in two steps: 1) Building single document strong chains while disambiguating the senses of the words and, 2) building multi-chain by merging the strongest chains of the single documents into one chain. The systems rank sentences using a formula that involves a) the lexical chain, b) keywords from query and c) named entities.

(Harabagiu et al., 2006) introduce a new paradigm for processing complex questions that relies on a combination of (a) question decompositions; (b) factoid QA techniques; and (c) Multi-Document Summarization (MDS) techniques. The question decomposition procedure operates on a Marcov chain, by following a random walk with mixture model on a bipartite graph of relations established between concepts related to the topic of a complex question and subquestions derived from topic-relevant passages that manifest these relations. Decomposed questions are then submitted to a state-of-the-art QA system in order to retrieve a set of passages that can later be merged into a comprehensive answer by a MDS system. They show that question decompositions using this method can significantly enhance the relevance and comprehensiveness of summary-length answers to complex questions.

There are approaches that are based on probabilistic models (Pingali et al., 2007) (Toutanova et al., 2007). (Pingali et al., 2007) rank the sentences based on a mixture model where each component of the model is a statistical model:

$$Score(s) = \alpha \times QIScore(s) + (1-\alpha) \times QFocus(s, Q)$$

Where, Score(s) is the score for sentence s. Query-independent score (QIScore) and query-dependent score (QFocus) are calculated based on probabilistic models. (Toutanova et al., 2007) learns a log-linear sentence ranking model by maximizing three metrics of sentence goodness: (a) ROUGE oracle, (b) Pyramid-derived, and (c) Model Frequency. The scoring function is learned by fitting weights for a set of feature functions of sentences in the document set and is trained to optimize a sentence pair-wise ranking criterion. The scoring function is further adapted to apply to summaries rather than sentences and to take into account redundancy among sentences.

There are approaches in "Recognizing Textual Entailment", "Sentence Alignment" and "Question Answering" that use syntactic and/or semantic information in order to measure the similarity between two textual units. (MacCartney et al., 2006) use typed dependency graphs (same as dependency trees) to represent the text and the hypothesis. Then they try to find a good partial alignment between the typed dependency graphs representing the hypothesis and the text in a search space of $O((m + 1)n)$

where hypothesis graph contains n nodes and a text graph contains m nodes. (Hirao et al., 2004) represent the sentences using Dependency Tree Path (DTP) to incorporate syntactic information. They apply String Subsequence Kernel (SSK) to measure the similarity between the DTPs of two sentences. They also introduce Extended String Subsequence Kernel (ESK) to incorporate semantics in DTPs. (Kouylekov and Magnini, 2005) use the tree edit distance algorithms on the dependency trees of the text and the hypothesis to recognize the textual entailment. According to this approach, a text T entails a hypothesis H if there exists a sequence of transformations (i.e. deletion, insertion and substitution) applied to T such that we can obtain H with an overall cost below a certain threshold. (Punyakanok et al., 2004) represent the question and the sentence containing answer with their dependency trees. They add semantic information (i.e. named entity, synonyms and other related words) in the dependency trees. They apply the approximate tree matching in order to decide how similar any given pair of trees are. They also use the edit distance as the matching criteria in the approximate tree matching. All these methods show the improvement over the BOW scoring methods.

Our Basic Element (BE)-based feature used the dependency tree to extract the BEs (i.e. head-modifier-relation) and ranked the BEs based on their log-likelihood ratios. For syntactic feature, we extracted the syntactic trees for the sentence as well as for the query using the Charniak parser and measured the similarity between the two trees using the tree kernel function. We used the ASSERT semantic role labeler system to parse the sentence as well as the query semantically and used the shallow semantic tree kernel to measure the similarity between the two shallow-semantic trees.

3 Feature Extraction

The sentences in the document collection are analyzed in various levels and each of the document-sentences is represented as a vector of feature-values. The features can be divided into several categories:

3.1 Lexical Features

3.1.1 N-gram Overlap

N-gram overlap measures the overlapping word sequences between the candidate sentence and the query sentence. With the view to measure the N-gram (N=1,2,3,4) overlap scores, a *query pool* and a *sentence pool* are created. In order to create the query (or sentence) pool, we took the query (or document) sentence and created a set of related sentences by replacing its important words[1] by their first-sense synonyms. For example given a stemmed document-sentence: "John write a poem", the sentence pool contains: "John compose a poem", "John write a verse form" along with the given sentence. We measured the recall based n-gram scores for a sentence P using the following formula:

n-gramScore(P) = $max_i(max_j$ N-gram$(s_i, q_j))$

$$\text{N-gram(S,Q)} = \frac{\sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{gram_n \in S} Count(gram_n)}$$

Where, n stands for the length of the n-gram $(n = 1, 2, 3, 4)$ and $Count_{match}\,(gram_n)$ is the number of n-grams co-occurring in the query and the candidate sentence, q_j is the j^{th} sentence in the query pool and s_i is the i^{th} sentence in the sentence pool of sentence P.

3.1.2 LCS, WLCS and Skip-Bigram

A sequence $W = [w_1, w_2, ..., w_n]$ is a subsequence of another sequence $X = [x_1, x_2, ..., x_m]$, if there exists a strict increasing sequence $[i_1, i_2, ..., i_k]$ of indices of X such that for all $j = 1, 2, ..., k$ we have $x_{i_j} = w_j$. Given two sequences, S_1 *and* S_2, the Longest Common Subsequence (LCS) of S_1 *and* S_2 is a common subsequence with maximum length. The longer the LCS of two sentences is, the more similar the two sentences are.

The basic LCS has a problem that it does not differentiate LCSes of different spatial relations within their embedding sequences (Lin, 2004). To improve the basic LCS method, we can remember the length of consecutive matches encountered so far to a regular two dimensional dynamic program table computing LCS. We call this weighted LCS (WLCS) and use k to indicate the length of the current consecutive matches ending at words x_i and y_j. Given two sentences X and Y, the WLCS score of X and Y can be computed using the similar dynamic programming procedure as stated in (Lin, 2004). We computed the LCS and WLCS-based F-measure following (Lin, 2004) using both the query pool and the sentence pool as in the previous section.

Skip-bigram is any pair of words in their sentence order, allowing for arbitrary gaps. Skip-bigram measures the overlap of skip-bigrams between a candidate sentence and a query sentence. Following (Lin, 2004), we computed the skip bi-gram score using both the sentence pool and the query pool.

[1]hence forth important words are the nouns, verbs, adverbs and adjectives

3.1.3 Head and Head Related-words Overlap

The number of head words common in between two sentences can indicate how much they are relevant to each other. In order to extract the heads from the sentence (or query), the sentence (or query) is parsed by Minipar [2] and from the dependency tree we extract the heads which we call exact head words. For example, the head word of the sentence: "John eats rice" is "eat".

We take the synonyms, hyponyms and hypernyms[3] of both the query-head words and the sentence-head words and form a set of words which we call head-related words. We measured the exact head score and the head-related score as follows:

$$\text{ExactHeadScore} = \frac{\sum_{w_1 \in HeadSet} Count_{match}(w_1)}{\sum_{w_1 \in HeadSet} Count(w_1)}$$

$$\text{HeadRelatedScore} = \frac{\sum_{w_1 \in HeadRelSet} Count_{match}(w_1)}{\sum_{w_1 \in HeadRelSet} Count(w_1)}$$

Where HeadSet is the set of head words in the sentence and $Count_{match}$ is the number of matches between the HeadSet of the query and the sentence. HeadRelSet is the set of synonyms, hyponyms and hypernyms of head words in the sentence and $Count_{match}$ is the number of matches between the head-related words of the query and the sentence.

3.2 Lexical Semantic Features

We form a set of words which we call *QueryRelatedWords* by taking the important words from the query, their first-sense synonyms, the nouns' hypernyms/hyponyms and important words from the nouns' gloss definitions.

Synonym overlap measure is the overlap between the list of synonyms of the important words extracted from the candidate sentence and the *QueryRelatedWords*. *Hypernym/hyponym overlap* measure is the overlap between the list of hypernyms and hyponyms of the nouns extracted from the sentence and the *QueryRelatedWords*, and *gloss overlap* measure is the overlap between the list of important words that are extracted from the gloss definitions of the nouns of the sentence and the *QueryRelatedWords*.

[2]http://www.cs.ualberta.ca/ lindek/minipar.htm

[3]hypernym and hyponym levels are restricted to 2 and 3 respectively

3.3 Statistical Similarity Measures

Statistical similarity measures are based on the co-occurance of similar words in a corpus. We have used two statistical similarity measures: *1. Dependency-based similarity measure* and *2. Proximity-based similarity measure*.

Dependency-based similarity measure uses the dependency relations among words in order to measure the similarity. It extracts the dependency triples then uses statistical approach to measure the similarity. *Proximity-based similarity measure* is computed based on the linear proximity relationship between words only. It uses the information theoretic definition of similarity to measure the similarity.

We used the data provided by Dr. Dekang Lin[4]. Using the data, one can retrieve most similar words for a given word. The similar words are grouped into clusters. Note that, for a word there can be more than one cluster. Each cluster represents the sense of the word and its similar words for that sense.

For each query word, we extract all of its clusters from the data. Now, in order to determine the right cluster for a query word, we measure the overlap score between the *QueryRelatedWords* and the *clusters of words*. The hypothesis is that, the cluster that has more words common with the *QueryRelatedWords* is the right cluster. We chose the cluster for a word which has the highest overlap score.

Once we get the clusters for the query words, we measured the overlap between the cluster words and the sentence words as follows:

$$\text{Measure} = \frac{\sum_{w_1 \in SenWords} Count_{match}(w_1)}{\sum_{w_1 \in SenWords} Count(w_1)}$$

Where, SenWords is the set of important words extracted from the sentence and $Count_{match}$ is the number of matches between the sentence words and the clusters of similar words of the query words.

3.4 Graph-based Similarity Measure

In LexRank (Erkan and Radev, 2004), the concept of graph-based centrality is used to rank a set of sentences, in producing generic multi-document summaries. A similarity graph is produced for the sentences in the document collection. In the graph, each node represents a sentence. The edges between the nodes measure the cosine similarity between the respective pair of sentences. The degree of a given node is an indication of how much important the sentence is. Once the similarity graph is

[4]http://www.cs.ualberta.ca/ lindek/downloads.htm

constructed, the sentences are then ranked according to their eigenvector centrality. To apply LexRank to query-focused context, a topic-sensitive version of LexRank is proposed in (Otterbacher et al., 2005). We followed a similar approach in order to calculate this feature. The score of a sentence is determined by a mixture model of the relevance of the sentence to the query and the similarity of the sentence to other high-scoring sentences.

3.5 Syntactic and Semantic Features:

So far, we have included the features of type Bag of Words (BOW). The task like *query-based summarization* that requires the use of more complex syntactic and semantics, the approaches with only BOW are often inadequate to perform fine-level textual analysis. We extracted three features that incorporate syntactic/semantic information.

3.5.1 Basic Element (BE) Overlap Measure

The "head-modifier-relation" triples, extracted from the dependency trees are considered as BEs in our experiment. The triples encode some syntactic/semantic information and one can quite easily decide whether any two units match or not- considerably more easily than with longer units (Zhou et al., 2005). We used the BE package distributed by ISI[5] to extract the BEs for the sentences.

Once we get the BEs for a sentence, we computed the Likelihood Ratio (LR) for each BE following (Zhou et al., 2005). Sorting BEs according to their LR scores produced a BE-ranked list. Our goal is to generate a summary that will answer the user questions. The ranked list of BEs in this way contains important BEs at the top which may or may not be relevant to the user questions. We filter those BEs by checking whether they contain any word which is a *query word* or a *QueryRelatedWords* (defined in Section 3.2). The score of a sentence is the sum of its BE scores divided by the number of BEs in the sentence.

3.5.2 Syntactic Feature

Encoding syntactic structure is easier and straight forward. Given a sentence (or query), we first parse it into a syntactic tree using a syntactic parser (i.e. Charniak parser) and then we calculate the similarity between the two trees using the *tree kernel* defined in (Collins and Duffy, 2001).

3.5.3 Shallow-semantic Feature

Though introducing BE and syntactic information gives an improvement on BOW by the use of dependency/syntactic parses, but these, too are not adequate when dealing with complex questions whose answers are expressed by long and articulated sentences or even paragraphs. Shallow semantic representations, bearing a more compact information, could prevent the sparseness of deep structural approaches and the weakness of BOW models (Moschitti et al., 2007).

[5]BE website:http://www.isi.edu/ cyl/BE

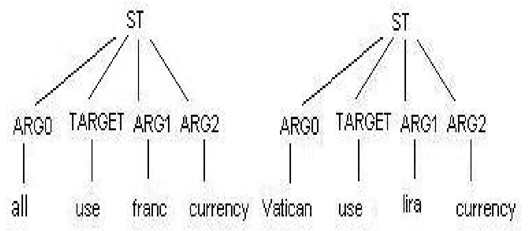


Figure 1: Example of semantic trees

Initiatives such as PropBank (PB) (Kingsbury and Palmer, 2002) have made possible the design of accurate automatic Semantic Role Labeling (SRL) systems like ASSERT (Hacioglu et al., 2003). For example, consider the PB annotation:

[ARG0 all][TARGET use][ARG1 the french franc][ARG2 as their currency]

Such annotation can be used to design a shallow semantic representation that can be matched against other semantically similar sentences, e.g.

[ARG0 the Vatican][TARGET use][ARG1 the Italian lira][ARG2 as their currency]

In order to calculate the semantic similarity between the sentences, we first represent the annotated sentence using the tree structures like Figure 1 which we call Semantic Tree (ST). In the semantic tree, arguments are replaced with the most important word-often referred to as the semantic head.

The sentences may contain one or more subordinate clauses. For example the sentence, "the Vatican, located wholly within Italy uses the Italian lira as their currency." gives the STs as in Figure 2. As we can see in Figure 2(A), when an argument node corresponds to an entire subordinate clause, we label its leaf with ST, e.g. the leaf of ARG0. Such ST node is actually the root of the subordinate clause in Figure 2(B). If taken separately, such STs do not express the whole meaning of the sentence, hence it is more accurate to define a single structure encoding the dependency between the two predicates as in Figure 2(C). We refer to this kind of nested STs as STNs.

Note that, the tree kernel (TK) function defined in (Collins and Duffy, 2001) computes the number of common subtrees between two trees. Such subtrees are subject to the constraint that their nodes are taken with all or none of the children they have in the original tree.

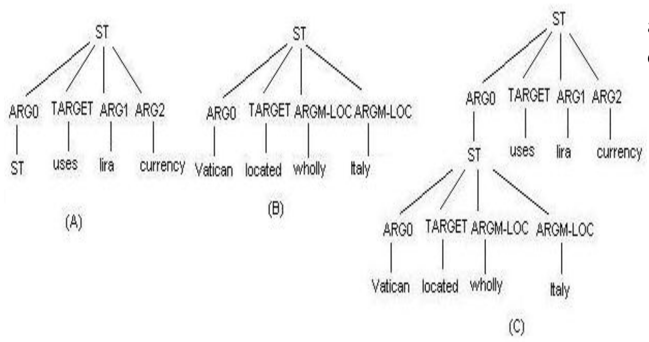


Figure 2: Two STs composing a STN

Though, this definition of subtrees makes the TK function appropriate for syntactic trees but at the same time makes it not well suited for the semantic trees (ST) defined above. For instance, although the two STs of Figure 1 share most of the subtrees rooted in the *ST* node, the kernel defined above computes only one match (ST ARG0 TARGET ARG1 ARG2) which is not useful.

The critical aspect of the TK function is that the productions of two evaluated nodes have to be identical to allow the match of further descendants. This means that common substructures cannot be composed by a node with only some of its children as an effective ST representation would require. (Moschitti et al., 2007) solve this problem by designing the Shallow Semantic Tree Kernel (SSTK) which allows to match portions of a ST. We followed the similar approach to compute the SSTK.

4 Ranking Sentences

In this section, we describe the scoring techniques in detail.

4.1 Learning Feature-weights: A Local Search Strategy

In order to fine-tune the weights of the features, we used a local search technique with simulated annealing to find the global maximum. Initially, we set all the feature-weights, $w_1, \cdots, w_n$, as equal values (i.e. 0.5) (see Algorithm 1). Based on the current weights we score the sentences and generate summaries accordingly. We evaluate the summaries using the automatic evaluation tool ROUGE (Lin, 2004) (described in Section 6) and the ROUGE value works as the feedback to our learning loop. Our learning system tries to maximize the ROUGE score in every step by changing the weights individually by a specific step size (i.e. 0.01). That means, to learn weight w_i, we change the value of w_i keeping all other weight values ($w_j \forall_{j \neq i}$) stagnant. For each weight w_i, the algorithm achieves the local maximum of ROUGE value. In order to find the global maximum we ran this algorithm multiple times with different random choices of initial values (i.e. simulated annealing).

```
**Input**: Stepsize $l$, Weight Initial Value $v$
**Output**: A vector $\vec{w}$ of learned weights
Initialize the weight values $w_i$ to $v$.
**for** $i \leftarrow 1 \; to \; n$ **do**
    $rg1 = rg2 = prev = 0$
    **while** *(true)* **do**
        scoreSentences($\vec{w}$)
        generateSummaries()
        $rg2$ = evaluateROUGE()
        **if** $rg1 \leq rg2$ **then**
            prev = $w_i$
            $w_i + = l$
            $rg1 = rg2$
        **else**
            break
        **end**
    **end**
**end**
return $\vec{w}$
```

Algorithm 1: Tuning weights using Local Search technique

Once we have learned the feature-weights, our *empirical* method computes the final scores for the sentences using the formula:

$$score_i = \vec{x_i}.\vec{w} \quad (1)$$

Where, $\vec{x_i}$ is the feature vector for i-th sentence, $\vec{w}$ is the weight vector and $score_i$ is the score of i-th sentence.

4.2 K-means Learning

We start with a set of initial cluster centers and go through several iterations of assigning each object to the cluster whose center is closest. After all objects have been assigned, we recompute the center of each cluster as the centroid or mean (μ) of its members.

Once we have learned the means of the clusters using the k-means algorithm, our next task is to rank the sentences according to a probability model. We have used Bayesian model in order to do so. Bayes' law says:

$$P(q_k|\vec{x}, \Theta) = \frac{p(\vec{x}|q_k, \Theta)P(q_k|\Theta)}{\sum_{k=1}^{K} p(\vec{x}|q_k, \Theta)p(q_k|\Theta)} \quad (2)$$

where q_k is a class, $\vec{x}$ is a feature vector representing a sentence and Θ is the parameter set of all class models. We set the weights of the clusters as equiprobable (i.e. $P(q_k|\Theta) = 1/K$). We calculated

$p(x|q_k, \Theta)$ using the gaussian probability distribution. The gaussian probability density function (pdf) for the d-dimensional random variable $\vec{x}$ is given by:

$$p_{(\mu,\Sigma)}(\vec{x}) = \frac{e^{\frac{-1}{2}(\vec{x}-\mu)^T \Sigma^{-1}(\vec{x}-\mu)}}{\sqrt{2\pi}^d \sqrt{det(\Sigma)}} \quad (3)$$

where μ, the mean vector and Σ, the covariance matrix are the parameters of the gaussian distribution. We get the means (μ) from the k-means algorithm and we calculate the covariance matrix using the unbiased covariance estimation:

$$\hat{\Sigma} = \frac{1}{N-1} \sum_{i=1}^{N} (x_j - \mu_j)(x_i - \mu_i)^T \quad (4)$$

4.3 EM Learning

EM is an iterative two step procedure: 1. Expectation-step and 2. Maximization-step. In the expectation step, we compute expected values for the hidden variables $h_{i,j}$ which are cluster membership probabilities. Given the current parameters, we compute how likely an object belongs to any of the clusters. The maximization step computes the most likely parameters of the model given the cluster membership probabilities. The data-points are considered to be generated by a mixture model of k-gaussians of the form:

$$P(\vec{x}) = \sum_{i=1}^{k} P(C=i) P(\vec{x}|\mu_i, \Sigma_i) \quad (5)$$

Where the total likelihood of model Θ with k components given the observed data points, $X = x_1, \cdots, x_n$ is:

$$\begin{aligned} L(\Theta|X) &= \prod_{i=1}^{n} \sum_{j=1}^{k} P(C=j) P(x_i|\Theta_j) \\ &= \prod_{i=1}^{n} \sum_{j=1}^{k} w_j P(x_i|\mu_j, \Sigma_j) \\ &\Leftrightarrow \sum_{i=1}^{n} log \sum_{j=1}^{k} w_j P(x_i|\mu_j, \Sigma_j) \end{aligned}$$

where P is the probability density function (i.e. eq 3). μ_j and Σ_j are the mean and covariance matrix of component j, respectively. Each component contributes a proportion, w_j, of the total population, such that: $\sum_{j=1}^{K} w_j = 1$.

However, a significant problem with the EM algorithm is that it converges to a local maximum of the likelihood function and hence the quality of the result depends on the initialization. In order to get good results from using random starting values, we can run the EM algorithm several times and choose the initial configuration for which we get the maximum log likelihood among all configurations. Choosing the best one among several runs is very computer intensive process. So, to improve the outcome of the EM algorithm on gaussian mixture models it is necessary to find a better method of estimating initial means for the components. To achieve this aim we explored the widely used "k-means" algorithm as a cluster (means) finding method. That means, the means found by k-means clustering above will be utilized as the initial means for EM and we calculate the initial covariance matrices using the unbiased covariance estimation procedure (eq:4).

Once the sentences are clustered by EM algorithm, we filter out the sentences which are not query-relevant by checking their probabilities, $P(q_r|x_i, \Theta)$ where, q_r denotes the cluster "query-relevant". If for a sentence x_i, $P(q_r|x_i, \Theta) > 0.5$ then x_i is considered to be query-relevant.

Our next task is to rank the query-relevant sentences in order to include them in the summary. This can be done easily by multiplying the feature vector $\vec{x_i}$ with the weight vector $\vec{w}$ that we learned by the local search technique (eq:1).

5 Redundancy Checking

When many of the competing sentences are included in the summary, the issue of information overlap between parts of the output comes up, and a mechanism for addressing redundancy is needed. Therefore, our summarization systems employ a final level of analysis: before being added to the final output, the sentences deemed to be important are compared to each other and only those that are not too similar to other candidates are included in the final answer or summary. Following (Zhou et al., 2005), we modeled this by BE overlap between an intermediate summary and a to-be-added candidate summary sentence. We call this overlap ratio R, where R is between 0 and 1 inclusively. Setting $R = 0.7$ means that a candidate summary sentence, s, can be added to an intermediate summary, S, if the sentence has a BE overlap ratio less than or equal to 0.7.

6 Experimental Evaluation

6.1 Evaluation Setup

We used the main task of Document Understanding Conference (DUC) 2007 for evaluation. The task was: *"Given a complex question (topic description) and a collection of relevant documents, the task is to synthesize a fluent, well-organized 250-word summary of the documents that answers the question(s) in the topic."*

NIST assessors developed topics of interest to them and choose a set of 25 documents relevant (document cluster) to each topic. Each topic and its document cluster were given to 4 different NIST assessors. The assessor created a 250-word summary of the document cluster that satisfies the information need expressed in the topic statement. These multiple "reference summaries" are used in the evaluation of summary content.

We carried out automatic evaluation of our summaries using ROUGE (Lin, 2004) toolkit, which has been widely adopted by DUC for automatic summarization evaluation. It measures summary quality by counting overlapping units such as the n-grams (ROUGE-N), word sequences (ROUGE-L and ROUGE-W) and word pairs (ROUGE-S and ROUGE-SU) between the candidate summary and the reference summary. ROUGE parameters were set as the same as DUC 2007 evaluation setup.

One purpose of our experiments is to study the impact of different features for complex question answering task. To accomplish this, we generated summaries for the topics of DUC 2007 by each of our seven systems defined as below:

The **LEX** system generates summaries based on only lexical features: n-gram (n=1,2,3,4), LCS, WLCS, skip bi-gram, head, head synonym. The **LSEM** system considers only lexical semantic features: synonym, hypernym/hyponym, gloss, dependency-based and proximity-based similarity. The **COS** system generates summary based on the graph-based method. The **SYS1** system considers all the features except the BE, syntactic and semantic features. The **SYS2** system considers all the features except the syntactic and semantic features. The **SYS3** considers all the features except the semantic and the **ALL**[6] system generates summaries taking all the features into account.

6.2 Evaluation Results

Table 1[7] to Table 3, Table 4 to Table 6 and Table 7 to Table 9 show the evaluation measures for k-means, EM and empirical approaches respectively. As Table 1 shows, in k-means, SYS2 gets 0-21%, SYS3 gets 4-32% and ALL gets 3-36% improvement in *ROUGE-2* scores over the SYS1 system. We get best *ROUGE-W* (Table 2) scores for SYS2 (i.e. including BE) but SYS3 and ALL do not perform well in this case. SYS2 improves the ROUGE-W F-score by 1% over SYS1. We do not get any improvement in *ROUGE-SU* (Table 3) scores when we include any kind of syntactic/semantic structures.

The case is different for EM and empirical approaches. Here, in every case we get a significant amount of improvement when we include the syntactic and/or semantic features. For EM (Table 4 to Table 6), the ratio of improvement in F-scores over SYS1 is: 1-3% for SYS2, 3-15% for SYS3 and 2-24% for ALL. In our empirical approach (Table 7 to Table 9), SYS2, SYS3 and ALL improve the F-scores by 3-11%, 7-15% and 8-19% over SYS1 respectively. These results clearly indicate the positive impact of the syntactic/semantic features for complex question answering task.

Score	LEX	LSEM	COS	SYS1	SYS2	SYS3	ALL
R	0.074	0.077	0.086	0.075	0.075	0.078	0.077
P	0.081	0.084	0.093	0.081	0.098	0.107	0.110
F	0.078	0.080	0.089	0.078	0.085	0.090	0.090

Table 1: ROUGE-2 measures in k-means learning

Table 10 shows the F-scores of the ROUGE measures for one baseline system, the best system in DUC 2007 and our three scoring techniques considering all features. The baseline system gener-

[6]SYS2, SYS3 and ALL systems show the impact of BE, syntactic and semantic features respectively

[7]R stands for Recall, P stands for Precision and F stands for F-score

Score	LEX	LSEM	COS	SYS1	SYS2	SYS3	ALL
R	0.098	0.097	0.101	0.099	0.101	0.097	0.097
P	0.195	0.194	0.200	0.237	0.233	0.241	0.237
F	0.130	0.129	0.134	0.140	0.141	0.139	0.138

Table 2: ROUGE-W measures in k-means learning

Score	LEX	LSEM	COS	SYS1	SYS2	SYS3	ALL
R	0.131	0.127	0.139	0.136	0.135	0.135	0.135
P	0.155	0.152	0.162	0.176	0.171	0.174	0.174
F	0.142	0.139	0.150	0.153	0.151	0.152	0.152

Table 3: ROUGE-SU in k-means learning

Score	LEX	LSEM	COS	SYS1	SYS2	SYS3	ALL
R	0.089	0.080	0.087	0.085	0.085	0.089	0.091
P	0.096	0.087	0.094	0.092	0.095	0.116	0.138
F	0.092	0.083	0.090	0.088	0.090	0.101	0.109

Table 4: ROUGE-2 measures in EM learning

Score	LEX	LSEM	COS	SYS1	SYS2	SYS3	ALL
R	0.103	0.096	0.101	0.102	0.101	0.102	0.101
P	0.205	0.193	0.200	0.203	0.218	0.222	0.223
F	0.137	0.128	0.134	0.136	0.138	0.139	0.139

Table 5: ROUGE-W measures in EM learning

Score	LEX	LSEM	COS	SYS1	SYS2	SYS3	ALL
R	0.146	0.128	0.138	0.143	0.144	0.145	0.144
P	0.171	0.153	0.162	0.168	0.177	0.186	0.185
F	0.157	0.140	0.149	0.154	0.159	0.163	0.162

Table 6: ROUGE-SU measures in EM learning

Score	LEX	LSEM	COS	SYS1	SYS2	SYS3	ALL
R	0.086	0.080	0.087	0.087	0.090	0.095	0.099
P	0.093	0.087	0.094	0.094	0.112	0.115	0.116
F	0.089	0.083	0.090	0.090	0.100	0.104	0.107

Table 7: ROUGE-2 in empirical approach

Score	LEX	LSEM	COS	SYS1	SYS2	SYS3	ALL
R	0.102	0.096	0.101	0.102	0.102	0.104	0.105
P	0.203	0.193	0.200	0.204	0.239	0.246	0.247
F	0.135	0.128	0.134	0.137	0.143	0.147	0.148

Table 8: ROUGE-W in empirical approach

Score	LEX	LSEM	COS	SYS1	SYS2	SYS3	ALL
R	0.144	0.129	0.138	0.145	0.146	0.149	0.150
P	0.169	0.153	0.162	0.171	0.182	0.195	0.197
F	0.155	0.140	0.150	0.157	0.162	0.169	0.170

Table 9: ROUGE-SU in empirical approach

ates summaries by returning all the leading sentences (up to 250 words) in the $\langle TEXT \rangle$ field of the most recent document(s). It shows that the empirical approach outperforms the other two learning techniques and EM performs better than k-means algorithm. EM improves the F-scores over k-means by 0.7-22.5%. Empirical approach improves the F-scores over k-means and EM by 5.9-20.2% and 3.5-6.5% respectively. Comparing with the DUC 2007 participants our systems achieve top scores and for some ROUGE measures there is no statistically significant difference between our system and the best DUC 2007 system.

System	ROUGE-1	ROUGE-2	ROUGE-W	ROUGE-SU
Baseline	0.335	0.065	0.114	0.113
Best	0.438	0.122	0.153	0.174
k-means	0.390	0.090	0.138	0.152
EM	0.399	0.109	0.139	0.162
Empirical	0.413	0.107	0.148	0.170

Table 10: F-measures for different systems

7 Conclusion and Future Work

Our experiments show the following: (a) our approaches achieve promising results, (b) empirical approach outperforms the other two learning and EM performs better than the k-means algorithm for this particular task, and (c) our systems achieve better results when we include BE, syntactic and semantic features.

In future, we have the plan to decompose the complex questions into several simple questions before measuring the similarity between the document sentence and the query sentence. We expect that by decomposing complex questions into the sets of subquestions that they entail, systems can improve the average quality of answers returned and achieve better coverage for the question as a whole.

References

M. Collins and N. Duffy. 2001. Convolution Kernels for Natural Language. In *Proceedings of Neural Information Processing Systems*, pages 625–632, Vancouver, Canada.

G. Erkan and D. R. Radev. 2004. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

K. Hacioglu, S. Pradhan, W. Ward, J. H. Martin, and D. Jurafsky. 2003. Shallow Semantic Parsing Using Support Vector Machines. In *Technical Report TR-CSLR-2003-03*, University of Colorado.

S. Harabagiu, F. Lacatusu, and A. Hickl. 2006. Answering complex questions with random walk models. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 220 – 227. ACM.

T. Hirao, , J. Suzuki, H. Isozaki, and E. Maeda. 2004. Dependency-based sentence alignment for multiple document summarization. In *Proceedings of Coling 2004*, pages 446–452, Geneva, Switzerland. COLING.

P. Kingsbury and M. Palmer. 2002. From Treebank to PropBank. In *Proceedings of the international conference on Language Resources and Evaluation*, Las Palmas, Spain.

M. Kouylekov and B. Magnini. 2005. Recognizing textual entailment with tree edit distance algorithms. In *Proceedings of the PASCAL Challenges Workshop: Recognising Textual Entailment Challenge*.

J. Li, L. Sun, C. Kit, and J. Webster. 2007. A Query-Focused Multi-Document Summarizer Based on Lexical Chains. In *Proceedings of the Document Understanding Conference*, Rochester. NIST.

C. Y. Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proceedings of Workshop on Text Summarization Branches Out, Post-Conference Workshop of Association for Computational Linguistics*, pages 74–81, Barcelona, Spain.

B. MacCartney, T. Grenager, M.C. de Marneffe, D. Cer, and C. D. Manning. 2006. Learning to recognize features of valid textual entailments. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL*, page 4148, New York, USA.

A. Moschitti, S. Quarteroni, R. Basili, and S. Manandhar. 2007. Exploiting Syntactic and Shallow Semantic Kernels for Question/Answer Classificaion. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 776–783, Prague, Czech Republic. ACL.

J. Otterbacher, G. Erkan, and D. R. Radev. 2005. Using Random Walks for Question-focused Sentence Retrieval. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 915–922, Vancouver, Canada.

P. Pingali, Rahul K., and V. Varma. 2007. IIIT Hyderabad at DUC 2007. In *Proceedings of the Document Understanding Conference*, Rochester. NIST.

V. Punyakanok, D. Roth, and W. Yih. 2004. Mapping dependencies trees: An application to question answering. In *Proceedings of AI & Math*, Florida, USA.

K. Toutanova, C. Brockett, M. Gamon, J. Jagarlamudi, H. Suzuki, and L. Vanderwende. 2007. The PYTHY Summarization System: Microsoft Research at DUC 2007 . In *proceedings of the Document Understanding Conference*, Rochester. NIST.

D. Zhang and W. S. Lee. 2003. A Language Modeling Approach to Passage Question Answering. In *Proceedings of the Twelfth Text REtreival Conference*, pages 489–495, Gaithersburg, Maryland.

L. Zhou, C. Y. Lin, and E. Hovy. 2005. A BE-based Multi-dccument Summarizer with Query Interpretation. In *Proceedings of Document Understanding Conference*, Vancouver, B.C., Canada.

Sampling Alignment Structure under a Bayesian Translation Model

John DeNero, Alexandre Bouchard-Côté and Dan Klein
Computer Science Department
University of California, Berkeley
{denero, bouchard, klein}@cs.berkeley.edu

Abstract

We describe the first tractable Gibbs sampling procedure for estimating phrase pair frequencies under a probabilistic model of phrase alignment. We propose and evaluate two nonparametric priors that successfully avoid the degenerate behavior noted in previous work, where overly large phrases memorize the training data. Phrase table weights learned under our model yield an increase in BLEU score over the word-alignment based heuristic estimates used regularly in phrase-based translation systems.

1 Introduction

In phrase-based translation, statistical knowledge of translation equivalence is primarily captured by counts of how frequently various phrase pairs occur in training bitexts. Since bitexts do not come segmented and aligned into phrase pairs, these counts are typically gathered by fixing a word alignment and applying phrase extraction heuristics to this word-aligned training corpus. Alternatively, phrase pair frequencies can be learned via a probabilistic model of phrase alignment, but this approach has presented several practical challenges.

In this paper, we address the two most significant challenges in phrase alignment modeling. The first challenge is with inference: computing alignment expectations under general phrase models is #P-hard (DeNero and Klein, 2008). Previous phrase alignment work has sacrificed consistency for efficiency, employing greedy hill-climbing algorithms and constraining inference with word alignments (Marcu and Wong, 2002; DeNero et al., 2006; Birch et al., 2006). We describe a Gibbs sampler that consistently and efficiently approximates expectations, using only polynomial-time computable operators. Despite the combinatorial complexity of the phrase alignment space, our sampled phrase pair expectations are guaranteed to converge to the true posterior distributions under the model (in theory) and do converge to effective values (in practice).

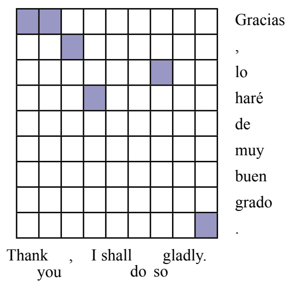
The second challenge in learning phrase alignments is avoiding a degenerate behavior of the general model class: as with many models which can choose between large and small structures, the larger structures win out in maximum likelihood estimation. Indeed, the maximum likelihood estimate of a joint phrase alignment model analyzes each sentence pair as one large phrase with no internal structure (Marcu and Wong, 2002). We describe two nonparametric priors that empirically avoid this degenerate solution.

Fixed word alignments are used in virtually every statistical machine translation system, if not to extract phrase pairs or rules directly, then at least to constrain the inference procedure for higher-level models. We estimate phrase translation features consistently using an inference procedure that is not constrained by word alignments, or any other heuristic. Despite this substantial change in approach, we report translation improvements over the standard word-alignment-based heuristic estimates of phrase table weights. We view this result as an important step toward building fully model-based translation systems that rely on fewer procedural heuristics.

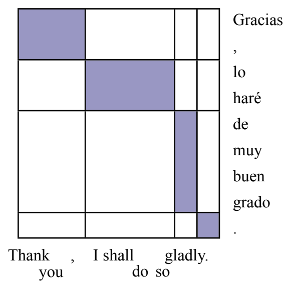
2 Phrase Alignment Model

While state-of-the-art phrase-based translation systems include an increasing number of features, translation behavior is largely driven by the phrase pair count ratios $\phi(e|f)$ and $\phi(f|e)$. These features are typically estimated heuristically using the counts $c(\langle e, f\rangle)$ of all phrase pairs in a training corpus that are licensed by word alignments:

$$\phi(e|f) = \frac{c(\langle e, f\rangle)}{\sum_{e'} c(\langle e', f\rangle)} \quad .$$



(a) example word alignment



(b) example phrase alignment

Figure 1: In this corpus example, the phrase alignment model found the non-literal translation pair $\langle$*gladly, de muy buen grado*$\rangle$ while heuristically-combined word alignment models did not. (a) is a *grow-diag-final-and* combined IBM Model 4 word alignment; (b) is a phrase alignment under our model.

In contrast, a generative model that explicitly aligns pairs of phrases $\langle e, f\rangle$ gives us well-founded alternatives for estimating phrase pair scores. For instance, we could use the model's parameters as translation features. In this paper, we compute the expected counts of phrase pairs in the training data according to our model, and derive features from these expected counts. This approach endows phrase pair scores with well-defined semantics relative to a probabilistic model. Practically, phrase models can discover high-quality phrase pairs that often elude heuristics, as in Figure 1. In addition, the model-based approach fits neatly into the framework of statistical learning theory for unsupervised problems.

2.1 Generative Model Description

We first describe the symmetric joint model of Marcu and Wong (2002), which we will extend. A two-step generative process constructs an ordered set of English phrases $e_{1:m}$, an ordered set of foreign phrases $f_{1:n}$, and a phrase-to-phrase alignment between them, $\mathbf{a} = \{(j, k)\}$ indicating that $\langle e_j, f_k\rangle$ is an aligned pair.

1. Choose a number of components ℓ and generate each of ℓ phrase pairs independently.

2. Choose an ordering for the phrases in the foreign language; the ordering for English is fixed by the generation order.[1]

[1]We choose the foreign to reorder without loss of generality.

In this process, $m = n = |\mathbf{a}|$; all phrases in both sentences are aligned one-to-one.

We parameterize the choice of ℓ using a geometric distribution, denoted P_{G}, with stop parameter $p_{\$}$:

$$P(\ell) = P_{\text{G}}(\ell; p_{\$}) = p_{\$} \cdot (1 - p_{\$})^{\ell-1} \ .$$

Each aligned phrase pair $\langle e, f\rangle$ is drawn from a multinomial distribution θ_{J} which is unknown. We fix a simple distortion model, setting the probability of a permutation of the foreign phrases proportional to the product of position-based distortion penalties for each phrase:

$$\begin{aligned} P(\mathbf{a}|\{\langle e, f\rangle\}) &\propto \prod_{a \in \mathbf{a}} \delta(a) \\ \delta(a = (j, k)) &= b^{|pos(e_j) - pos(f_k) \cdot s|} \ , \end{aligned}$$

where $pos(\cdot)$ denotes the word position of the start of a phrase, and s the ratio of the length of the English to the length of the foreign sentence. This positional distortion model was deemed to work best by Marcu and Wong (2002).

We can now state the joint probability for a phrase-aligned sentence consisting of ℓ phrase pairs:

$$P(\{\langle e, f\rangle\}, \mathbf{a}) = P_{\text{G}}(\ell; p_{\$}) P(\mathbf{a}|\{\langle e, f\rangle\}) \prod_{\langle e, f\rangle} \theta_{\text{J}}(\langle e, f\rangle) \ .$$

While this model has several free parameters in addition to θ_{J}, we fix them to reasonable values to focus learning on the phrase pair distribution.[2]

2.2 Unaligned Phrases

Sentence pairs do not always contain equal information on both sides, and so we revise the generative story to include unaligned phrases in both sentences. When generating each component of a sentence pair, we first decide whether to generate an aligned phrase pair or, with probability $p_{\o}$, an unaligned phrase.[3] Then, we either generate an aligned phrase pair from θ_{J} or an unaligned phrase from θ_{N}, where θ_{N} is a multinomial over phrases. Now, when generating $e_{1:m}$, $f_{1:n}$ and alignment $\mathbf{a}$, the number of phrases $m + n$ can be greater than $2 \cdot |\mathbf{a}|$.

[2]Parameters were chosen by hand during development on a small training corpus. $p_{\$} = 0.1, b = 0.85$ in experiments.

[3]We strongly discouraged unaligned phrases in order to align as much of the corpus as possible: $p_{\o} = 10^{-10}$ in experiments.

To unify notation, we denote unaligned phrases as phrase pairs with one side equal to *null*: $\langle e, null\rangle$ or $\langle null, f\rangle$. Then, the revised model takes the form:

$$\begin{aligned} P(\{\langle e,f\rangle\},\mathbf{a}) &= P_{\mathrm{G}}(\ell; p_{\$})P(\mathbf{a}|\{\langle e,f\rangle\})\prod_{\langle e,f\rangle} P_{\mathrm{M}}(\langle e,f\rangle) \\ P_{\mathrm{M}}(\langle e,f\rangle) &= p_{\o}\theta_{\mathrm{N}}(\langle e,f\rangle) + (1-p_{\o})\theta_{\mathrm{J}}(\langle e,f\rangle)\,. \end{aligned}$$

In this definition, the distribution θ_{N} gives non-zero weight only to unaligned phrases of the form $\langle e, null\rangle$ or $\langle null, f\rangle$, while θ_{J} gives non-zero weight only to aligned phrase pairs.

3 Model Training and Expectations

Our model involves observed sentence pairs, which in aggregate we can call x, latent phrase segmentations and alignments, which we can call z, and parameters θ_{J} and θ_{N}, which together we can call θ. A model such as ours could be used either for the learning of the key phrase pair parameters in θ, or to compute expected counts of phrase pairs in our data. These two uses are very closely related, but we focus on the computation of phrase pair expectations. For exposition purposes, we describe a Gibbs sampling algorithm for computing expected counts of phrases under $P(z|x, \theta)$ for fixed θ. Such expectations would be used, for example, to compute maximum likelihood estimates in the E-step of EM. In Section 4, we instead compute expectations under $P(z|x)$, with θ marginalized out entirely.

In a Gibbs sampler, we start with a *complete* phrase segmentation and alignment, state z_0, which sets all latent variables to some initial configuration. We then produce a sequence of sample states z_i, each of which differs from the last by some small local change. The samples z_i are guaranteed (in the limit) to consistently approximate the conditional distribution $P(z|x, \theta)$ (or $P(z|x)$ later). Therefore, the average counts of phrase pairs in the samples converge to expected counts under the model. Normalizing these expected counts yields estimates for the features $\phi(e|f)$ and $\phi(f|e)$.

Gibbs sampling is not new to the natural language processing community (Teh, 2006; Johnson et al., 2007). However, it is usually used as a search procedure akin to simulated annealing, rather than for approximating expectations (Goldwater et al., 2006; Finkel et al., 2007). Our application is also atypical for an NLP application in that we use an approximate sampler not only to include Bayesian prior information (section 4), but also because computing phrase alignment expectations exactly is a #P-hard problem (DeNero and Klein, 2008). That is, we could not run EM exactly, even if we wanted maximum likelihood estimates.

3.1 Related Work

Expected phrase pair counts under $P(z|x, \theta)$ have been approximated before in order to run EM. Marcu and Wong (2002) employed local search from a heuristic initialization and collected alignment counts during a hill climb through the alignment space. DeNero et al. (2006) instead proposed an exponential-time dynamic program pruned using word alignments. Subsequent work has relied heavily on word alignments to constrain inference, even under reordering models that admit polynomial-time E-steps (Cherry and Lin, 2007; Zhang et al., 2008).

None of these approximations are consistent, and they offer no method of measuring their biases. Gibbs sampling is not only consistent in the limit, but also allows us to add Bayesian priors conveniently (section 4). Of course, sampling has liabilities as well: we do not know in advance how long we need to run the sampler to approximate the desired expectations "closely enough."

Snyder and Barzilay (2008) describe a Gibbs sampler for a bilingual morphology model very similar in structure to ours. However, the basic sampling step they propose – resampling all segmentations and alignments for a sequence at once – requires a #P-hard computation. While this asymptotic complexity was apparently not prohibitive in the case of morphological alignment, where the sequences are short, it is prohibitive in phrase alignment, where the sentences are often very long.

3.2 Sampling with the SWAP Operator

Our Gibbs sampler repeatedly applies each of five operators to each position in each training sentence pair. Each operator freezes all of the current state z_i except a small local region, determines all the ways that region can be reconfigured, and then chooses a (possibly) slightly different z_{i+1} from among those outcomes according to the conditional probability of each, given the frozen remainder of the state. This frozen region of the state is called a *Markov blanket* (denoted m), and plays a critical role in proving the correctness of the sampler.

The first operator we consider is SWAP, which changes alignments but not segmentations. It freezes the set of phrases, then picks two English phrases e_1 and e_2 (or two foreign phrases, but we focus on the English case). All alignments are frozen except the phrase pairs $\langle e_1, f_1\rangle$ and $\langle e_2, f_2\rangle$. SWAP chooses between keeping $\langle e_1, f_1\rangle$ and $\langle e_2, f_2\rangle$ aligned as they are (outcome o_0), or swapping their alignments to create $\langle e_1, f_2\rangle$ and $\langle e_2, f_1\rangle$ (outcome o_1).

SWAP chooses stochastically in proportion to each outcome's posterior probability: $P(o_0|m, x, \theta)$ and $P(o_1|m, x, \theta)$. Each phrase pair in each outcome contributes to these posteriors the probability of adding a new pair, deciding whether it is null, and generating the phrase pair along with its contribution to the distortion probability. This is all captured in a succinct potential function $\psi(\langle e, f\rangle) =$

$$\begin{cases} (1-p_\$)\,(1-p_\varnothing)\,\theta_\mathrm{J}(\langle e, f\rangle)\,\delta(\langle e, f\rangle) & e \,\&\, f \text{ non-}\mathit{null} \\ (1-p_\$)\cdot p_\varnothing \cdot \theta_\mathrm{N}(\langle e, f\rangle) & \text{otherwise} \end{cases}$$

.

Thus, outcome o_0 is chosen with probability $P(o_0|m, x, \theta) =$

$$\frac{\psi(\langle e_1, f_1\rangle)\psi(\langle e_2, f_2\rangle)}{\psi(\langle e_1, f_1\rangle)\psi(\langle e_2, f_2\rangle) + \psi(\langle e_1, f_2\rangle)\psi(\langle e_2, f_1\rangle)}.$$

Operators in a Gibbs sampler require certain conditions to guarantee the correctness of the sampler. First, they must choose among *all possible configurations* of the unfrozen local state. Second, immediately re-applying the operator from any outcome must yield the same set of outcome options as before.[4] If these conditions are not met, the sampler may no longer be guaranteed to yield consistent approximations of the posterior distribution.

A subtle issue arises with SWAP as defined: should it also consider an outcome o_2 of $\langle e_1, \mathit{null}\rangle$ and $\langle e_2, \mathit{null}\rangle$ that removes alignments? No part of the frozen state is changed by removing these alignments, so the first Gibbs condition dictates that we must include o_2. However, after choosing o_2, when we reapply the operator to positions e_1 and e_2, we freeze all alignments except $\langle e_1, \mathit{null}\rangle$ and $\langle e_2, \mathit{null}\rangle$, which prevents us from returning to o_0. Thus, we fail to satisfy the second condition. This point is worth emphasizing because some prior work has treated Gibbs sampling as randomized search and, intentionally or otherwise, proposed inconsistent operators.

[4]These are two sufficient conditions to guarantee that the Metropolis-Hastings acceptance ratio of the sampling step is 1.

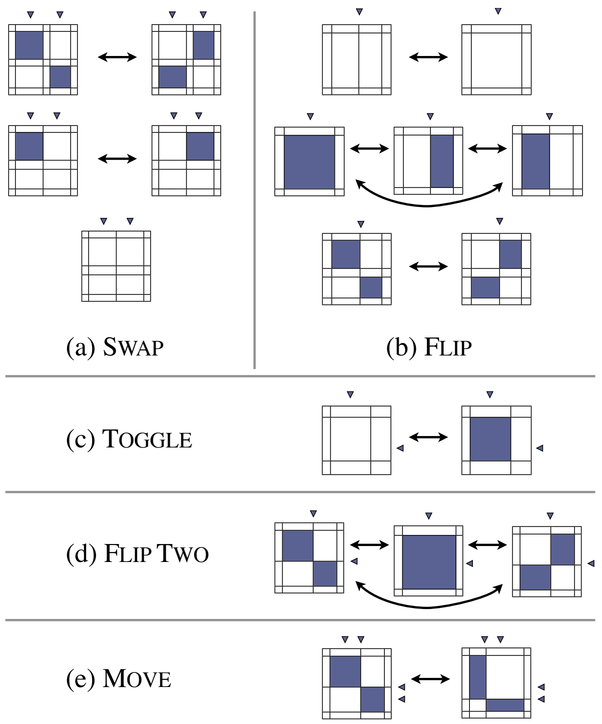


Figure 2: Each local operator manipulates a small portion of a single alignment. Relevant phrases are exaggerated for clarity. The outcome sets (depicted by arrows) of each possible configuration are fully connected. Certain configurations cannot be altered by certain operators, such as the final configuration in SWAP. Unalterable configurations for TOGGLE have been omitted for space.

Luckily, the problem is not with SWAP, but with our justification of it: we can salvage SWAP by augmenting its Markov blanket. Given that we have selected $\langle e_1, f_1\rangle$ and $\langle e_2, f_2\rangle$, we not only freeze all other alignments and phrase boundaries, but also the number of aligned phrase pairs. With this count held invariant, o_2 is not among the possible outcomes of SWAP given m. Moreover, regardless of the outcome chosen, SWAP can immediately be reapplied at the same location with the same set of outcomes.

All the possible starting configurations and outcome sets for SWAP appear in Figure 2(a).

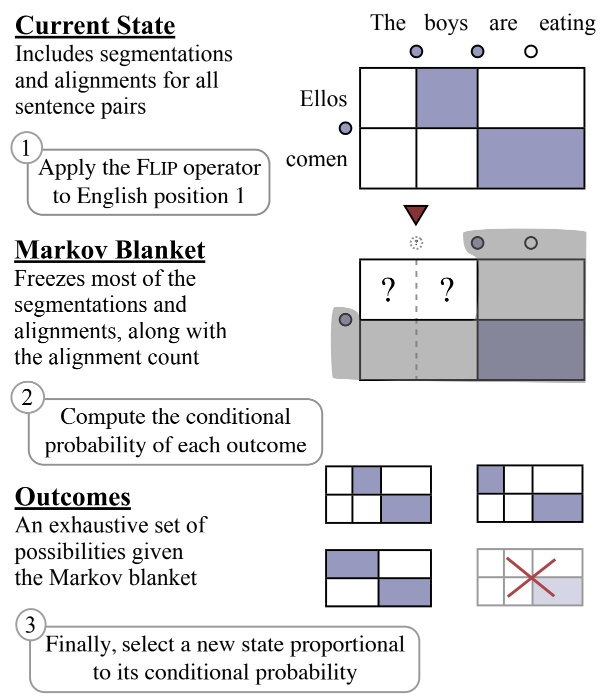

Current State
Includes segmentations and alignments for all sentence pairs
The boys are eating
Ellos
comen
1 Apply the FLIP operator to English position 1
Markov Blanket
Freezes most of the segmentations and alignments, along with the alignment count
? ?
2 Compute the conditional probability of each outcome
Outcomes
An exhaustive set of possibilities given the Markov blanket
3 Finally, select a new state proportional to its conditional probability

Figure 3: The three steps involved in applying the FLIP operator. The Markov blanket freezes all segmentations except English position 1 and all alignments except those for *Ellos* and *The boys*. The blanket also freezes the number of alignments, which disallows the lower right outcome.

3.3 The FLIP operator

SWAP can arbitrarily shuffle alignments, but we need a second operator to change the actual phrase boundaries. The FLIP operator changes the status of a single *segmentation position*[5] to be either a phrase boundary or not. In this sense FLIP is a bilingual analog of the segmentation boundary flipping operator of Goldwater et al. (2006).

Figure 3 diagrams the operator and its Markov blanket. First, FLIP chooses any between-word position in either sentence. The outcome sets for FLIP vary based on the current segmentation and adjacent alignments, and are depicted in Figure 2.

Again, for FLIP to satisfy the Gibbs conditions, we must augment its Markov blanket to freeze not only all other segmentation points and alignments, but also the number of aligned phrase pairs. Otherwise, we end up allowing outcomes from which we cannot return to the original state by reapplying FLIP. Consequently, when a position is already segmented and both adjacent phrases are currently aligned, FLIP cannot unsegment the point because it can't create two aligned phrase pairs with the one larger phrase that results (see bottom of Figure 2(b)).

[5]A segmentation position is a position between two words that is also potentially a boundary between two phrases in an aligned sentence pair.

3.4 The TOGGLE operator

Both SWAP and FLIP freeze the number of alignments in a sentence. The TOGGLE operator, on the other hand, can add or remove individual alignment links. In TOGGLE, we first choose an e_1 and f_1. If $\langle e_1, f_1 \rangle \in \mathbf{a}$ or both e_1 and f_1 are *null*, we freeze all segmentations and the rest of the alignments, and choose between including $\langle e_1, f_1 \rangle$ in the alignment or leaving both e_1 and f_1 unaligned. If only one of e_1 and f_1 are aligned, or they are not aligned to each other, then TOGGLE does nothing.

3.5 A Complete Sampler

Together, FLIP, SWAP and TOGGLE constitute a complete Gibbs sampler that consistently samples from the posterior $P(z|x, \theta)$. Not only are these operators valid Gibbs steps, but they also can form a path of positive probability from any source state to any target state in the space of phrase alignments (formally, the induced Markov chain is *irreducible*). Such a path can at worst be constructed by unaligning all phrases in the source state with TOGGLE, composing applications of FLIP to match the target phrase boundaries, then applying TOGGLE to match the target alignments.

We include two more local operators to speed up the rate at which the sampler explores the hypothesis space. In short, FLIP TWO simultaneously flips an English and a foreign segmentation point (to make a large phrase out of two smaller ones or vice versa), while MOVE shifts an aligned phrase boundary to the left or right. We omit details for lack of space.

3.6 Phrase Pair Count Estimation

With our sampling procedure in place, we can now estimate the expected number of times a given phrase pair occurs in our data, for fixed θ, using a Monte-Carlo average,

$$\frac{1}{N} \sum_{i=1}^{N} \text{count}_{\langle e,f \rangle}(x, z_i) \xrightarrow{a.s.} \mathbb{E}\left[\text{count}_{\langle e,f \rangle}(x, \cdot)\right] \quad .$$

The left hand side is simple to compute; we count aligned phrase pairs in each sample we generate. In practice, we only count phrase pairs after applying every operator to every position in every sentence (one iteration).[6] Appropriate normalizations of these expected counts can be used either in an M-step as maximum likelihood estimates, or to compute values for features $\phi(f|e)$ and $\phi(e|f)$.

4 Nonparametric Bayesian Priors

The Gibbs sampler we presented addresses the inference challenges of learning phrase alignment models. With slight modifications, it also enables us to include prior information into the model. In this section, we treat θ as a random variable and shape its prior distribution in order to correct the well-known degenerate behavior of the model.

4.1 Model Degeneracy

The structure of our joint model penalizes explanations that use many small phrase pairs. Each phrase pair token incurs the additional expense of generation and distortion. In fact, the maximum likelihood estimate of the model puts mass on $\langle e, f\rangle$ pairs that span entire sentences, explaining the training corpus with one phrase pair per sentence.

Previous phrase alignment work has primarily mitigated this tendency by constraining the *inference procedure*, for example with word alignments and linguistic features (Birch et al., 2006), or by disallowing large phrase pairs using a non-compositional constraint (Cherry and Lin, 2007; Zhang et al., 2008). However, the problem lies with the model, and therefore should be corrected in the model, rather than the inference procedure.

Model-based solutions appear in the literature as well, though typically combined with word alignment constraints on inference. A sparse Dirichlet prior coupled with variational EM was explored by Zhang et al. (2008), but it did not avoid the degenerate solution. Moore and Quirk (2007) proposed a new conditional model structure that does not cause large and small phrases to compete for probability mass. May and Knight (2007) added additional model terms to balance the cost of long and short derivations in a syntactic alignment model.

[6]For experiments, we ran the sampler for 100 iterations.

4.2 A Dirichlet Process Prior

We control this degenerate behavior by placing a Dirichlet process (DP) prior over θ_{J}, the distribution over aligned phrase pairs (Ferguson, 1973).

If we were to assume a maximum number K of phrase pair types, a (finite) Dirichlet distribution would be an appropriate prior. A draw from a K-dimensional Dirichlet distribution is a list of K real numbers in $[0, 1]$ that sum to one, which can be interpreted as a distribution over K phrase pair types.

However, since the event space of possible phrase pairs is in principle unbounded, we instead use a Dirichlet process. A draw from a DP is a *countably infinite* list of real numbers in $[0, 1]$ that sum to one, which we interpret as a distribution over a countably infinite list of phrase pair types.[7]

The Dirichlet distribution and the DP distribution have similar parameterizations. A K-dimensional Dirichlet can be parameterized with a *concentration parameter* $\alpha > 0$ and a *base distribution* $M_0 = (\mu_1, \ldots, \mu_{K-1})$, with $\mu_i \in (0, 1)$.[8] This parameterization has an intuitive interpretation: under these parameters, the average of independent samples from the Dirichlet will converge to M_0. That is, the average of the ith element of the samples will converge to μ_i. Hence, the base distribution M_0 characterizes the sample mean. The concentration parameter α only affects the variance of the draws.

Similarly, we can parameterize the Dirichlet process with a concentration parameter α (that affects only the variance) and a base distribution M_0 that determines the mean of the samples. Just as in the finite Dirichlet case, M_0 is simply a probability distribution, but now with countably infinite support: all possible phrase pairs in our case. In practice, we can use an unnormalized M_0 (a base measure) by appropriately rescaling α.

In our model, we select a base measure that strongly prefers shorter phrases, encouraging the model to use large phrases only when it has sufficient evidence for them. We continue the model:

[7]Technical note: to simplify exposition, we restrict the discussion to settings such as ours where the base measure of the DP has countable support.

[8]This parametrization is equivalent to the standard pseudo-counts parametrization of K positive real numbers. The bijection is given by $\alpha = \sum_{i=1}^{K} \tilde{\alpha}_i$ and $\mu_i = \tilde{\alpha}_i/\alpha$, where $(\tilde{\alpha}_1, \ldots, \tilde{\alpha}_K)$ are the pseudo-counts.

$$\begin{aligned}
\theta_{\mathrm{J}} &\sim DP(M_0, \alpha) \\
M_0(\langle e, f\rangle) &= [P_f(f) P_{\mathrm{WA}}(e|f) \cdot P_e(e) P_{\mathrm{WA}}(f|e)]^{\frac{1}{2}} \\
P_f(f) &= P_{\mathrm{G}}(|f|; p_s) \cdot \left(\frac{1}{n_f}\right)^{|f|} \\
P_e(e) &= P_{\mathrm{G}}(|e|; p_s) \cdot \left(\frac{1}{n_e}\right)^{|e|} .
\end{aligned}$$

.

P_{WA} is the IBM model 1 likelihood of one phrase conditioned on the other (Brown et al., 1994). P_f and P_e are uniform over types for each phrase length: the constants n_f and n_e denote the vocabulary size of the foreign and English languages, respectively, and P_{G} is a geometric distribution.

Above, θ_{J} is drawn from a DP centered on the geometric mean of two joint distributions over phrase pairs, each of which is composed of a monolingual unigram model and a lexical translation component. This prior has two advantages. First, we pressure the model to use smaller phrases by increasing p_s ($p_s = 0.8$ in experiments). Second, we encourage good phrase pairs by incorporating IBM Model 1 distributions. This use of word alignment distributions is notably different from lexical weighting or word alignment constraints: we are supplying prior knowledge that phrases will generally follow word alignments, though with enough corpus evidence they need not (and often do not) do so in the posterior samples. The model proved largely insensitive to changes in the sparsity parameter α, which we set to 100 for experiments.

4.3 Unaligned phrases and the DP Prior

Introducing unaligned phrases invites further degenerate megaphrase behavior: a sentence pair can be generated cheaply as two unaligned phrases that each span an entire sentence. We attempted to place a similar DP prior over θ_{N}, but surprisingly, this modeling choice invoked yet another degenerate behavior. The DP prior imposes a *rich-get-richer* property over the phrase pair distribution, strongly encouraging the model to reuse existing pairs rather than generate new ones. As a result, common words consistently aligned to *null*, even while suitable translations were present, simply because each null alignment reinforced the next. For instance, *the* was always unaligned.

Instead, we fix θ_{N} to a simple unigram model that is uniform over word types. This way, we discourage unaligned phrases while focusing learning on θ_{J}. For simplicity, we reuse $P_f(f)$ and $P_e(e)$ from the prior over θ_{J}.

$$\theta_{\mathrm{N}}(\langle e, f\rangle) = \begin{cases} \frac{1}{2} \cdot P_e(e) & \text{if } f = \mathit{null} \\ \frac{1}{2} \cdot P_f(f) & \text{if } e = \mathit{null} \end{cases} .$$

The $\frac{1}{2}$ represents a choice of whether the aligned phrase is in the foreign or English sentence.

4.4 Collapsed Sampling with a DP Prior

Our entire model now has the general form $P(x, z, \theta_{\mathrm{J}})$; all other model parameters have been fixed. Instead of searching for a suitable θ_{J},[9] we sample from the posterior distribution $P(z|x)$ with θ_{J} marginalized out.

To this end, we convert our Gibbs sampler into a collapsed Gibbs sampler[10] using the Chinese Restaurant Process (CRP) representation of the DP (Aldous, 1985). With the CRP, we avoid the problem of explicitely representing samples from the DP. CRP-based samplers have served the community well in related language tasks, such as word segmentation and coreference resolution (Goldwater et al., 2006; Haghighi and Klein, 2007).

Under this representation, the probability of each sampling outcome is a simple expression in terms of the state of the rest of the training corpus (the Markov blanket), rather than explicitly using θ_{J}.

Let z_m be the set of aligned phrase pair tokens observed in the rest of the corpus. Then, when $\langle e, f\rangle$ is aligned (that is, neither e nor f are *null*), the conditional probability for a pair $\langle e, f\rangle$ takes the form:

$$\tau(\langle e, f\rangle | z_m) = \frac{\mathrm{count}_{\langle e,f\rangle}(z_m) + \alpha \cdot M_0(\langle e, f\rangle)}{|z_m| + \alpha},$$

where $\mathrm{count}_{\langle e,f\rangle}(z_m)$ is the number of times that $\langle e, f\rangle$ appears in z_m. We can write this expression thanks to the exchangeability of the model. For further exposition of this collapsed sampler posterior,

[9] For instance, using approximate MAP EM.

[10] A collapsed sampler is simply one in which the model parameters have been marginalized out.

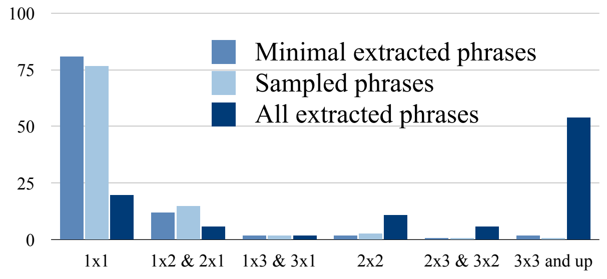


Figure 4: The distribution of phrase pair sizes (denoted *English length* x *foreign length*) favors small phrases under the model.

see Goldwater et al. (2006).[11]

The sampler remains exactly the same as described in Section 3, except that the posterior conditional probability of each outcome uses a revised potential function $\psi_{\text{DP}}(\langle e, f\rangle) =$

$$\begin{cases} (1-p_\$)\ (1-p_\emptyset)\ \tau(\langle e,f\rangle)\ \delta(\langle e,f\rangle) & e\ \&\ f \text{ non-}null \\ (1-p_\$)\cdot p_\emptyset \cdot \theta_{\text{N}}(\langle e,f\rangle) & \text{otherwise} \ . \end{cases}$$

ψ_{DP} is like ψ, but the fixed θ_{J} is replaced with the constantly-updated τ function.

4.5 Degeneracy Analysis

Figure 4 shows a histogram of phrase pair sizes in the distribution of expected counts under the model. As reference, we show the size distribution of both *minimal* and *all* phrase pairs extracted from word alignments using the standard heuristic. Our model tends to select minimal phrases, only using larger phrases when well motivated.[12]

This result alone is important: a model-based solution with no inference constraint has yielded a non-degenerate distribution over phrase lengths. Note that our sampler does find the degenerate solution quickly under a uniform prior, confirming that the model, and not the inference procedure, is selecting these small phrases.

[11]Note that the expression for τ changes slightly under conditions where two phrase pairs being changed simultaneously coincidentally share the same lexical content. Details of these fringe conditions have been omitted for space, but were included in our implementation.

[12]The largest phrase pair found was 13 English words by 7 Spanish words.

4.6 A Hierarchical Dirichlet Process Prior

We also evaluate a hierarchical Dirichlet process (HDP) prior over θ_{J}, which draws monolingual distributions θ_{E} and θ_{F} from a DP and θ_{J} from their cross-product:

$$\begin{aligned} \theta_{\text{J}} &\sim DP(M_0', \alpha) \\ M_0'(\langle e,f\rangle) &= [\theta_{\text{F}}(f) P_{\text{WA}}(e|f)\cdot \theta_{\text{E}}(e) P_{\text{WA}}(f|e)]^{\frac{1}{2}} \\ \theta_{\text{F}} &\sim DP(P_f, \alpha') \\ \theta_{\text{E}} &\sim DP(P_e, \alpha')\ . \end{aligned}$$

This prior encourages novel phrase pairs to be composed of phrases that have been used before. In the sampler, we approximate table counts for θ_{E} and θ_{F} with their expectations, which can be computed from phrase pair counts (see the appendix of Goldwater et al. (2006) for details). The HDP prior gives a similar distribution over phrase sizes.

5 Translation Results

We evaluate our new estimates using the baseline translation pipeline from the 2007 Statistical Machine Translation Workshop shared task.

5.1 Baseline System

We trained Moses on all Spanish-English Europarl sentences up to length 20 (177k sentences) using GIZA++ Model 4 word alignments and the *grow-diag-final-and* combination heuristic (Koehn et al., 2007; Och and Ney, 2003; Koehn, 2002), which performed better than any alternative combination heuristic.[13] The baseline estimates (*Heuristic*) come from extracting phrases up to length 7 from the word alignment. We used a bidirectional lexicalized distortion model that conditions on both foreign and English phrases, along with their orientations. Our 5-gram language model was trained on 38.3 million words of Europarl using Kneser-Ney smoothing. We report results with and without lexical weighting, denoted *lex*.

We tuned and tested on development corpora for the 2006 translation workshop. The parameters for each phrase table were tuned separately using minimum error rate training (Och, 2003). Results are

[13]Sampling iteration time scales quadratically with sentence length. Short sentences were chosen to speed up our experiment cycle.

Estimate	Phrase Pair Count	NIST BLEU	Exact Match METEOR
Heuristic	4.4M	29.8	52.4
DP	0.6M	28.8	51.7
HDP	0.3M	29.1	52.0
DP-composed	3.7M	30.1	52.7
HDP-composed	3.1M	30.1	52.6
DP-smooth	4.8M	30.1	52.5
HDP-smooth	4.6M	**30.2**	**52.7**
Heuristic + lex	4.4M	30.5	52.9
DP-smooth + lex	4.8M	30.4	53.0
HDP-smooth + lex	4.6M	**30.7**	**53.2**

Table 1: BLEU results for learned distributions improve over a heuristic baseline. Estimate labels are described fully in section 5.3. The label *lex* indicates the addition of a lexical weighting feature.

scored with lowercased, tokenized NIST BLEU, and exact match METEOR (Papineni et al., 2002; Lavie and Agarwal, 2007).

The baseline system gives a BLEU score of 29.8, which increases to 30.5 with *lex*, as shown in Table 1. For reference, training on all sentences of length less than 40 (the shared task baseline default) gives 32.4 BLEU with *lex*.

5.2 Learned Distribution Performance

We *initialized* the sampler with a configuration derived from the word alignments generated by the baseline. We greedily constructed a phrase alignment from the word alignment by identifying minimal phrase pairs consistent with the word alignment in each region of the sentence. We then ran the sampler for 100 iterations through the training data. Each iteration required 12 minutes under the DP prior, and 30 minutes under the HDP prior. Total running time for the HDP model neared two days on an eight-processor machine with 16 Gb of RAM.

Estimating phrase counts under the DP prior decreases BLEU to 28.8, or 29.1 under the HDP prior. This gap is not surprising: heuristic extraction discovers many more phrase pairs than sampling. Note that sacrificing only 0.7 BLEU while shrinking the phrase table by 92% is an appealing trade-off in resource-constrained settings.

5.3 Increasing Phrase Pair Coverage

The estimates *DP-composed* and *HDP-composed* in Table 1 take expectations of a more liberal count function. While sampling, we count not only aligned phrase pairs, but also larger ones composed of two or more contiguous aligned pairs. This count function is similar to the phrase pair extraction heuristic, but never includes unaligned phrases in any way. Expectations of these composite phrases still have a probabilistic interpretation, but they are not the structures we are directly modeling. Notably, these estimates outperform the baseline by 0.3 BLEU without ever extracting phrases from word alignments, and performance increases despite a reduction in table size.

We can instead increase coverage by smoothing the learned estimates with the heuristic counts. The estimates *DP-smooth* and *HDP-smooth* add counts extracted from word alignments to the sampler's running totals, which improves performance by 0.4 BLEU over the baseline. This smoothing balances the lower-bias sampler counts with the lower-variance heuristics ones.

6 Conclusion

Our novel Gibbs sampler and nonparametric priors together address two open problems in learning phrase alignment models, approximating inference consistently and efficiently while avoiding degenerate solutions. While improvements are modest relative to the highly developed word-alignment-centered baseline, we show for the first time competitive results from a system that uses word alignments only for model initialization and smoothing, rather than inference and estimation. We view this milestone as critical to eventually developing a clean probabilistic approach to machine translation that unifies model structure across both estimation and decoding, and decreases the use of heuristics.

References

David Aldous. 1985. Exchangeability and related topics. In *École d'été de probabilitiés de Saint-Flour*, Berlin. Springer.

Alexandra Birch, Chris Callison-Burch, and Miles Osborne. 2006. Constraining the phrase-based, joint probability statistical translation model. In *The Con-*

ference for the Association for Machine Translation in the Americas.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1994. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19:263–311.

Colin Cherry and Dekang Lin. 2007. Inversion transduction grammar for joint phrasal translation modeling. In *The Annual Conference of the North American Chapter of the Association for Computational Linguistics Workshop on Syntax and Structure in Statistical Translation*.

John DeNero and Dan Klein. 2008. The complexity of phrase alignment problems. In *The Annual Conference of the Association for Computational Linguistics: Short Paper Track*.

John DeNero, Dan Gillick, James Zhang, and Dan Klein. 2006. Why generative phrase models underperform surface heuristics. In *The Annual Conference of the North American Chapter of the Association for Computational Linguistics Workshop on Statistical Machine Translation*.

Thomas S Ferguson. 1973. A bayesian analysis of some nonparametric problems. In *Annals of Statistics*.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. 2007. The infinite tree. In *The Annual Conference of the Association for Computational Linguistics*.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *The Annual Conference of the Association for Computational Linguistics*.

Aria Haghighi and Dan Klein. 2007. Unsupervised coreference resolution in a nonparametric bayesian model. In *The Annual Conference of the Association for Computational Linguistics*.

Mark Johnson, Thomas Griffiths, and Sharon Goldwater. 2007. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *The Annual Conference of the Association for Computational Linguistics*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *The Annual Conference of the Association for Computational Linguistics*.

Philipp Koehn. 2002. Europarl: A multilingual corpus for evaluation of machine translation.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *The Annual Conference of the Association for Computational Linguistics Workshop on Statistical Machine Translation*.

Daniel Marcu and Daniel Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *The Conference on Empirical Methods in Natural Language Processing*.

Jonathan May and Kevin Knight. 2007. Syntactic realignment models for machine translation. In *The Conference on Empirical Methods in Natural Language Processing*.

Robert Moore and Chris Quirk. 2007. An iteratively-trained segmentation-free phrase translation model for statistical machine translation. In *The Annual Conference of the Association for Computational Linguistics Workshop on Statistical Machine Translation*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29:19–51.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *The Annual Conference of the Association for Computational Linguistics*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *The Annual Conference of the Association for Computational Linguistics*.

Benjamin Snyder and Regina Barzilay. 2008. Unsupervised multilingual learning for morphological segmentation. In *The Annual Conference of the Association for Computational Linguistics*.

Yee Whye Teh. 2006. A hierarchical Bayesian language model based on Pitman-Yor processes. In *The Annual Conference of the Association for Computational Linguistics*.

Hao Zhang, Chris Quirk, Robert C. Moore, and Daniel Gildea. 2008. Bayesian learning of non-compositional phrases with synchronous parsing. In *The Annual Conference of the Association for Computational Linguistics*.

Improving Chinese Semantic Role Classification with Hierarchical Feature Selection Strategy

Weiwei Ding
Institute of Computational Linguistics
Peking University
Beijing, 100871, China
weiwei.ding.pku@gmail.com

Baobao Chang
Institute of Computational Linguistics
Peking University
Beijing, 100871, China
chbb@pku.edu.cn

Abstract

In recent years, with the development of Chinese semantically annotated corpus, such as Chinese Proposition Bank and Normalization Bank, the Chinese semantic role labeling (SRL) task has been boosted. Similar to English, the Chinese SRL can be divided into two tasks: semantic role identification (SRI) and classification (SRC). Many features were introduced into these tasks and promising results were achieved. In this paper, we mainly focus on the second task: SRC. After exploiting the linguistic discrepancy between numbered arguments and ARGMs, we built a semantic role classifier based on a hierarchical feature selection strategy. Different from the previous SRC systems, we divided SRC into three sub tasks in sequence and trained models for each sub task. Under the hierarchical architecture, each argument should first be determined whether it is a numbered argument or an ARGM, and then be classified into fine-gained categories. Finally, we integrated the idea of exploiting argument interdependence into our system and further improved the performance. With the novel method, the classification precision of our system is 94.68%, which outperforms the strong baseline significantly. It is also the state-of-the-art on Chinese SRC.

1 Introduction

Semantic Role labeling (SRL) was first defined in Gildea and Jurafsky (2002). The purpose of SRL task is to identify and classify the semantic roles of each predicate in a sentence. The semantic roles are marked and each of them is assigned a tag which indicates the type of the semantic relation with the related predicate. Typical tags include Agent, Patient, Source, etc. and some adjuncts such as Temporal, Manner, Extent, etc. Since the arguments can provide useful semantic information, the SRL is crucial to many natural language processing tasks, such as Question and Answering (Narayanan and Harabagiu 2004), Information Extraction (Surdeanu et al. 2003), and Machine Translation(Boas 2002). With the efforts of many researchers (Carreras and Màrquez 2004, 2005, Moschitti 2004, Pradhan et al 2005, Zhang et al 2007), different machine learning methods and linguistics resources are applied in this task, which has made SRL task progress fast.

Compared to the research on English, the research on Chinese SRL is still in its infancy stage. Previous work on Chinese SRL mainly focused on how to transplant the machine learning methods which has been successful with English, such as Sun and Jurafsky (2004), Xue and Palmer (2005) and Xue (2008). Sun and Jurafsky (2004) did the preliminary work on Chinese SRL without any large semantically annotated corpus of Chinese. They just labeled the predicate-argument structures of ten specified verbs to a small collection of Chinese sentences, and used Support Vector Machines to identify and classify the arguments. This paper made the first attempt on Chinese SRL and produced promising results. After the PropBank (Xue and Palmer 2003) was built, Xue and Palmer (2005) and Xue (2008) have produced more complete and systematic research on Chinese SRL.

Moschitti et al. (2005) has made some preliminary attempt on the idea of hierarchical semantic role labeling. However, without considerations on how to utilize the characteristics of linguistically similar semantic roles, the purpose of the hierarchical system is to simplify the classification process to make it less time consuming. So the hierarchical system in their paper performs a little worse than the traditional SRL systems, although it is more efficient.

Xue and Palmer (2004) did very encouraging work on the feature calibration of semantic role labeling. They found out that different features suited for different sub tasks of SRL, i.e. semantic role identification and classification. For semantic analysis, developing features that capture the right kind of information is crucial. Experiments on Chinese SRL (Xue and Palmer 2005, Xue 2008) reassured these findings.

In this paper, we mainly focus on the semantic role classification (SRC) process. With the findings about the linguistic discrepancy of different semantic role groups, we try to build a 2-step semantic role classifier with hierarchical feature selection strategy. That means, for different sub tasks, different models will be trained with different features. The purpose of this strategy is to capture the right kind of information of different semantic role groups. It is hard to do manual selection of features since there are too many feature templates which has been proven to be useful in SRC; so, we designed a simple feature selection algorithm to select useful features automatically from a large set of feature templates. With this hierarchical feature selection architecture, our system can outperform previous systems. The selected feature templates for each process of SRC can in turn reassure the existence of the linguistic discrepancy. At last, we also integrate the idea of exploiting argument interdependence to make our system perform better.

The rest of the paper is organized as follows. In section 2, the semantically annotated corpus - Chinese Propbank is discussed. The architecture of our method is described in section 3. The feature selection strategy is discussed in section 4. The settings of experiments can be found in section 5. The results of the experiments can be found in section 6, where we will try to make some linguistic explanations of the selected features. Section 7 is conclusions and future work.

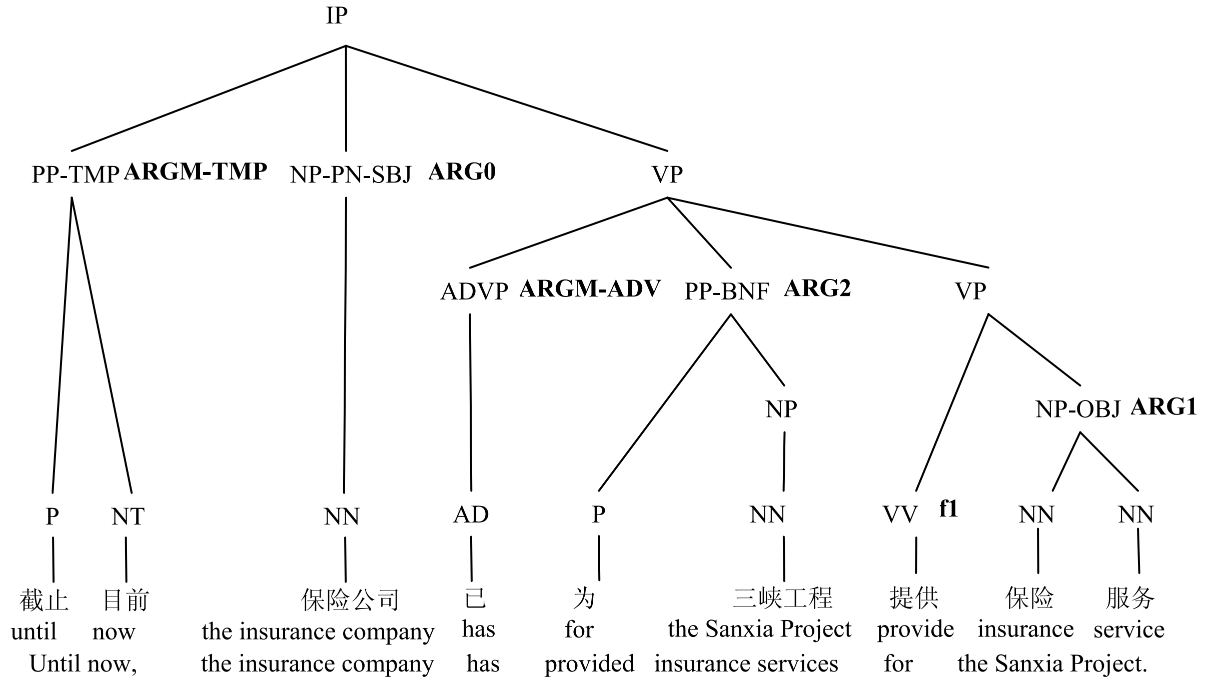

IP
PP-TMP ARGM-TMP
NP-PN-SBJ ARG0
VP
ADVP ARGM-ADV
PP-BNF ARG2
VP
NP
NP-OBJ ARG1
P NT NN AD P NN VV f1 NN NN
截止 目前 保险公司 已 为 三峡工程 提供 保险 服务
until now the insurance company has for the Sanxia Project provide insurance service
Until now, the insurance company has provided insurance services for the Sanxia Project.

Figure 1. an example from PropBank

2 The Chinese PropBank

The Chinese PropBank has labeled the predicate-argument structures of sentences from the Chinese TreeBank (Xue et al. 2005). It is constituted of two parts. One is the labeled data, which indicates the positions of the predicates and its arguments in the Chinese Treebank. The other is a dictionary which lists the frames of all the labeled predicates. Figure 1 is an example from the PropBank[1]. We put the word-by-word translation and the translation of the whole sentence below the example.

It is quite a complex sentence, as there are many semantic roles in it. In this sentence, all the semantic roles of the verb 提供 (provide) are presented in the syntactic tree. We can separate the semantic roles into two groups.

The first group of semantic roles can be called the core arguments, which capture the core relations. In this sentence, there are three arguments of verb 提供 (provide) in this sentence. 保险公司 (the insurance company) is labeled as ARG0, which is the proto-agent of the verb. Specifically to the verb 提供 (provide), it is the provider. 保险服务 (insurance services) is the direct object of the verb, and it is the proto-patient, which is labeled as ARG1. Specifically to the verb 提供 (provide), it represents things provided. 为三峡工程 (for the Sanxia Project) is another kind of argument, which is labeled as ARG1, and it represents the receiver.

The other group of semantic roles is called adjuncts. They are always used to reveal the peripheral information. There are two adjuncts of the target verb in this sentence: 截止目前 (until recently) and 已 (has), both of which are labeled as ARGM. However, the two ARGMs reveal information of different aspects. Besides the ARGM tags, the secondary tags "TMP" and "ADV" are assigned to the two semantic roles respectively. "TMP" indicates that 截止目前 (until recently) is a modifier representing the temporal information, and "ADV" indicates that 已 (has) is an adverbial modifier.

In the Chinese PropBank, the difference of the two groups is obvious. The core arguments are all labeled with numbers, and they are also called the numbered arguments. The numbers range from 0 to 4 in Chinese PropBank. The adjuncts are labeled with "ARGM".

3 Building a Hierarchical Semantic Role Classifier

In this section, we will discuss the linguistic fundaments of the construction of a hierarchical semantic role classifier. We use "hierarchical" to distinguish our classifier from the previous "flat" ones.

3.1 Linguistic Discrepancy of Different Semantic Role Groups

The purpose of the SRC task is to assign a tag to all the semantic roles which have been identified. The tags include ARG0-4, and 17 kinds of ARGMs (with functional tags). Previous SRC systems treat all the tags equally, and view the SRC as a multi-category classification task. However, we have different opinions of the traditional architecture.

Due to the discussions in section 2, we noticed that the semantic roles can be divided into two groups naturally according to the different kinds of semantic information represented by them. Here we will make some linguistic analysis of the two semantic role groups. Conversely to the process of the syntactic realization of semantic roles, we want to find out what linguistic features make a constituent ARG0 instead of ARG1, or another constituent ARGM-TMP instead of ARGM-ADV, i.e. what features capture the most crucial information of the two groups.

As what we have assumed, the linguistic features which made a syntactic constituent labeled as either one of the core arguments or one of the adjuncts varies greatly. Take the sentence in section 2 as an example, even if the only information we have about the phrase 截止目前 (until now) is that it is an adjunct of the verb, we can almost confirm, no matter where this node takes place in the parsing tree, this constituent will be labeled as ARGM-TMP. 已 (has) is also the same. According to its meaning, the only category can be assigned to it is ARGM-ADV. But, things are quite different to the core argument. In the same sentence, 保险公司 (the insurance company) is a good example. If we limit our observation to the phrase itself, we can hardly assert that it is the ARG0 of the target verb. Only when we extend our observation to the syntactic structure level, find out it is the subject of this sentence, and the voice of the sentence is active, the semantic type of 保险公司 (the insurance company) is finally confirmed. If we have another sentence in which 保险公司 (the insurance company) is not the subject, but rather the object, and the target verb is 开办 (set up), then it will probably be labeled as ARG1.

[1] This sentence is extracted from chtb_082.fid of Chinese PropBank 1.0, and we made some simplifications on it.

Due to the analysis above, we can conclude the linguistic discrepancy of the two semantic role groups as follows. Core arguments and adjuncts share different kinds of inner linguistic consistency respectively. For the core arguments, the specific type cannot be determined with the information of the arguments only. At this level, the core arguments are dependent on other information except the information about themselves. For example, the information of syntactic structures is crucial to the determination of the types of core arguments, and trivial differences of the syntactic structures will lead to the different output. Because of this, we can say that the core arguments are sensitive to the syntactic structures. Compared to the core arguments, adjuncts are the opposite. They are relatively independent on other information, since most of the adjuncts can be easily classified just based on the information about themselves[2]. And although the positions of the adjuncts in the syntactic structure can vary, the types of the adjuncts are fixed. In this sense, the adjuncts are insensitive to the syntactic structures.

After we made the linguistic discrepancy of the two semantic role groups, we can make a bold assumption that the differences of the two groups can be reflected in the capability of different kinds of features to capture the crucial information for the two groups. For example, the "voice" features seems to be crucial to the core arguments but useless to the adjuncts. This assumption provided us with the idea of a hierarchical feature selection system.

In this system, we first classify the constituents into two classes: core arguments and adjuncts. And then, the system classifies core arguments and adjuncts separately. For different subtasks we only select the most useful features and discard the less pertinent ones. We hope to take utilization of the most crucial features to improve semantic role classification.

3.2 System Architecture

Previous semantic role classifiers always did the classification problem in one-step. However, in this paper, we did SRC in two steps. The architectures of hierarchical semantic role classifiers can be found in figure 2, which is similar with that in Moschitti et al. (2005).

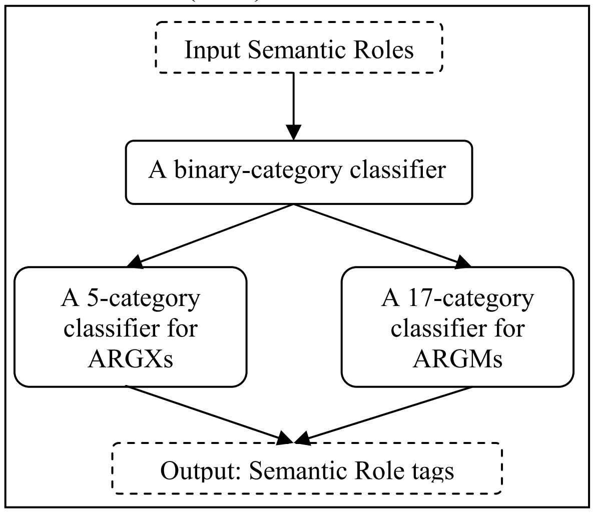

Input Semantic Roles
A binary-category classifier
A 5-category classifier for ARGXs
A 17-category classifier for ARGMs
Output: Semantic Role tags

Figure 2. The architecture of our hierarchical SRC system

As what has been shown in figure 2, a semantic role will first be determined whether it is a numbered argument or an ARGM by a binary-category classifier. And, then if the semantic role is a numbered argument, it will be determined by a 5-category classifier designed for ARGX, i.e. the numbered arguments. If it is an ARGM, the functional tag will be assigned by a 17-category classifier built for ARGMs. Accordingly, with this hierarchical architecture, the SRC problem is divided into 3 sub tasks, each of which has an independent classifier.

3.3 Integrating the Idea of Exploiting Argument Interdependence

Jiang et al. (2005) has built a semantic role classifier exploiting the interdependence of semantic roles. It has turned the single point classification problem into the sequence labeling problem with the introduction of semantic context features. Semantic context features indicates the features extracted from the arguments around the current one. We can use window size to represent the scope of the context. Window size [-m, n] means that, in the sequence that all the arguments has constructed, the features of previous m and following n arguments will be utilized for the classification of current semantic role. There are two kinds of argument sequences in Jiang et al. (2005), and we only test the linear sequence. Take the sentence in figure 1 as an example. The linear sequence of the arguments in this sentence is: 截止目前(until then),

[2] Extra features e.g. predicate may be still useful because that the information, provided by the high-level description of self-descriptive features, e.g. phrase type, are limited.

保险公司 (the insurance company), 已 (has), 为三峡工程 (for the Sanxia Project), 保险服务 (insurance services). For the argument 已 (has), if the semantic context window size is [-1,2], the semantic context features e.g. headword, phrase type and etc. of 保险公司 (the insurance company), 为三峡工程 (for the Sanxia Project) and 保险服务 (insurance services) will be utilized to serve the classification task of 已 (has).

While their paper has improved the SRC performance on English, it also has one potential disadvantage, which is that they didn't separate the core arguments and ARGMs. The influence and explanations of this defect are presented in Section 6. But in our hierarchical system, this problem can be solved. Since in the first step, we have separated the numbered arguments and ARGMs. We suppose that with the separation of the two semantic role groups, the system performance will be further improved.

4 Feature Selection Strategy

Due to what we have discussed in the section 3.1, we need to select different features for the three sub task of SRC. In this paper, we did not make the selection manually; however, we make a simple greedy strategy for feature selection to do it automatically. Although the best solution may not be found, automatic selection of features can try far more combinations of feature templates than manual selection. Because of this, this strategy possibly can produce a better local optional solution.

First, we built a pool of feature templates which has proven to be useful on the SRC. Most of the feature templates are standard, so only the new ones will be explained. The candidate feature templates include:

Voice from Sun and Jurafsky (2004).

Head word POS, Head Word of Prepositional Phrases, Constituent tree distance, from Pradhan etc. (2004).

Position, subcat frame, phrase type, first word, last word, subcat frame+, predicate, path, head word and its POS, predicate + head word, predicate + phrase type, path to BA and BEI, verb class[3]*, verb class + head word, verb class + phrase type,* from Xue (2008).

predicate POS, first word + last word, phrase type of the sibling to the left, phrase type of the sibling to the right, verb + subcate frame+, verb POS + subcat frame+, the amount of VPs in path, phrase type + phrase type of parent node, which can be easily understood by name.

voice position, indicates whether the voice marker (BA, BEI) is before or after the constituent in focus.

subcat frame,* the rule that expands the parent node of the constituent in focus.

subcat frame@, the rule that expands the constituent in focus.

layer of the constituent in focus, the number of constituents in the ascending part of the path subtracted by the number of those in the descending part of path, e.g. if the path is PP-BNF ↑ VP ↓ VP ↓ VV, the feature extracted by this template will be -1.

SemCat (semantic category) of predicate, SemCat of first word, SemCat of head word, SemCat of last word, SemCat of predicate + SemCat of first word, SemCat of predicate + SemCat of last word, predicate + SemCat of head word, SemCat of predicate + head word. The semantic categories of verbs and other words are extracted from the Semantic Knowledge-base of Contemporary Chinese (Wang et al. 2003).

verb AllFrameSets, the combination of all the framesets of a predicate.

verb class + verb AllFrameSets, verb AllFrameSets + head word, verb AllFrameSets + phrase type.

There are more than 40 feature templates, and it is quite difficult to traverse all the possible combinations and get the best one. So we use a greedy algorithm to get an approximate optimal solution.

The feature selection algorithm is as follows. Each time we choose one of the feature templates and add it into the system. The one, after which is added, the performance is the highest, will be chosen. Then we continue to choose feature templates until there are no one left. In the end, there are a series of feature sets, which recorded the process of feature selection. Then we choose the feature set which can perform the best on development set. The code of feature selection algorithm is designed in Figure 3.

[3] It is a bit different from Xue (2008), since we didn't use the syntactic alternation information.

```
1. add all feature templates to set $\mathbf{S}$, the set of
   selected feature templates $\mathbf{C_0}$ is null
2. for i = 0 to n-1, n is the number of elements
   in $\mathbf{S}$
3.     $\mathbf{P_i}$ = 0
4.     for each feature template $ft_j$ in set $\mathbf{S}$
5.         $\mathbf{C'_i} = \mathbf{C_i} + ft_j$
6.         train a model with features extracted
             by $\mathbf{C'_i}$ and test on development set
7.         if the result $\mathbf{P'} > \mathbf{P_i}$
8.             $\mathbf{P_i} = \mathbf{P'}$, k= j
9.     end for
10.    $\mathbf{C_{i+1}} = \mathbf{C_i} + ft_k$
11.    $\mathbf{S} = \mathbf{S} - ft_k$
12. end for
13. the set $\mathbf{C_m}$ correspondent to $\mathbf{P_m}$, which is
    the highest, will be chosen.
```

Figure 3. the greedy feature selection algorithm

To make a comparison, we also built a traditional 1-step semantic role classifier based on this feature selection strategy. We will take this classifier as the baseline system.

5 Experiment Settings

5.1 Classifier

In our SRL system, we use a Maximum Entropy toolkit with tunable Gaussian Prior and L-BFGS parameter estimation, which is implemented by Zhang Le. This toolkit is available at `http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html`. It can well handle the multi-category classification problem and it is quite efficient.

5.2 Data

We use Chinese PropBank 1.0 (LDC number: LDC2005T23) in our experiments. PropBank 1.0 includes the annotations for files chtb_001.fid to chtb_931.fid, or the first 250K words of the Chinese TreeBank 5.1. For the experiments, the data of PropBank is divided into three parts. 648 files (from chtb_081 to chtb_899.fid) are used as the training set. The development set includes 40 files, from chtb_041.fid to chtb_080.fid. The test set includes 72 files, which are chtb_001 to chtb_041, and chtb_900 to chtb_931. We use the same data setting with Xue (2008), however a bit different from Xue and Palmer (2005).

6 Results and Discussion

The results of the feature selection are presented in table1. In this table, "Baseline" indicates the 1-step architecture, and "Hierarchical" indicates the "hierarchical feature selection architecture" implemented in this paper. "X_M", "ARGX" and "ARGM" indicate the three sub-procedures of the hierarchical architecture, which are "ARGX and ARGM separation", "ARGX classification", "ARGM classification" respectively. "Y" in the table indicates that the feature template has been selected for the sub task.

According to table 1, we can find some interesting facts, which in turn prove what we found about semantic role groups in section 3.1.

In table 1, feature templates related to the syntactic structure includes: voice-related group (voice, voice information, path to BA and BEI), frame-related group (verb class, verb class + head word, verb class + phrase type, all frames of verb, verb class + all frames of verb), the layer of argument, position and 4 kinds of subcat frames. As we assumed before, these features are crucial to core arguments but of little use to adjuncts. The results have proven this assumption. Of the entire 14 syntactic structure-related feature templates, 8 were selected by the ARGX process but only 2 was selected by the ARGM process. The two exceptions should be viewed as the result of random impact, which cannot be avoided in automatic feature selection.

Compared with the different features selected by these tasks, we can find other interesting results. Few of the features selected by the X_M process also have related with the verb or the syntactic structures, which is quite similar with the ARGM process. This is probably because most of ARGMs are easy to be identified without syntactic structure information, which makes the opposite of ARGMs, i.e. the ARGXs easy to be filtered. Besides, the features selected by the baseline system have much in common with those selected by the ARGX process. This can be explained by the fact that both in the development and test set, the amount of core arguments outperforms that of adjuncts. The proportions between core arguments and adjuncts are 1.79:1 on the development set, and 1.63:1 on the test set. Because of the bias, the baseline system will tend to choose more syntactic structure-related features to label core arguments precisely.

Baseline	Hierarchical			Feature Name
	X_M	ARGX	ARGM	
		Y		predicate
Y		Y		predicate POS
	Y		Y	first word
			Y	first word + last word
Y		Y		head word
Y				head word POS
Y	Y			phrase type
	Y		Y	phrase type + phrase type of parent node
			Y	phrase type of the sibling to the left
Y		Y		phrase type of the sibling to the right
Y	Y			position
		Y		voice
Y				voice position
Y		Y		path to BA and BEI
Y	Y	Y		verb class
			Y	verb class + head word
Y	Y			verb class + phrase type
Y		Y		verb AllFrameSets
Y		Y		verb class + verb AllFrameSets
		Y		subcat frame
	Y			subcat frame*
		Y		subcate frame@
			Y	subcat frame+
Y		Y		layer of the constituent in focus
	Y	Y	Y	predicate + head word
Y	Y	Y	Y	predicate + phrase type
Y	Y	Y		SemCat of predicate
Y				SemCat of first word
Y		Y		SemCat of last word
			Y	SemCat of predicate + SemCat of last word
Y		Y		SemCat of head word

Table 1. Feature selection results for the baseline and the hierarchical system

	Baseline	Hierarchical
DEV	95.15%	**95.94%**
TEST	93.38%	**94.31%**

Table 2. Comparison of the performance between the baseline and hierarchical system

With this new architecture, we have achieved improvement on the performance of the semantic role classification, which can be found in table 2. Our classifier performs better both on the development and the test set. The labeled precision on the development set is from 95.15% to 95.94%, and the test set is from 93.38% to 94.31%, with an ERR (error reduction rate) of 14.05%. Both of the improvements are statistically significant ($\chi 2$ test with p= 0.05). The errors of SRC have three origins, which are correspondent to the three sub tasks of the hierarchical architecture. Detailed information of the comparison between the two systems can be found in table 3, which can tell us where the improvements come from.

	Baseline	Hierarchical
ARGX/ARGM errors	1.66%	1.75%
inner ARGX errors	3.59%	2.75%
inner ARGM errors	1.37%	1.19%
TOTAL	6.62%	5.69%

Table 3 Error rate analysis on the test set

In table 3, the percentages are calculated the way that the number of the errors was divided by the number of the arguments in the test set. ARGX/ARGM errors represent the errors that the semantic roles are classified into wrong group, e.g. ARGXs are labeled as ARGMs and vice versa. The inner errors represent the errors in a group, e.g. ARG0 are labeled as ARG1. From this table, we can find that ARGX is the most difficult task. X-M and ARGM are less challenging. Besides the relatively little error reduction in the ARGM process, the greatest part of improvement comes from the process of the most difficult sub task: the ARGX sub task. It is a bit surprising that the first step of the X_M in the hierarchical system process did not perform better than that in the baseline system.

	Baseline	Hierarchical	Sum
ARG0	96.14%	96.58%	2046
ARG1	92.75%	94.60%	2428
ARG2	78.46%	78.85%	260
ARG3	60.00%	76.00%	25
ARG4	40.00%	100.00%	5
ARGM-ADV	96.64%	96.85%	1490
ARGM-ASP	100.00%	0.00%	1
ARGM-BNF	91.30%	86.96%	23
ARGM-CND	77.78%	77.78%	9
ARGM-CRD	N/A	N/A	0
ARGM-DGR	N/A	N/A	0
ARGM-DIR	54.84%	58.06%	31
ARGM-DIS	79.38%	79.38%	97
ARGM-EXT	50.00%	25.00%	8
ARGM-FRQ	N/A	N/A	0
ARGM-LOC	90.91%	92.21%	308
ARGM-MNR	89.92%	91.13%	248
ARGM-PRD	N/A	N/A	0
ARGM-PRP	97.83%	97.83%	46
ARGM-TMP	95.41%	96.30%	675
ARGM-TPC	33.33%	8.33%	12
TBERR[4]	0.00%	0.00%	2

Table 4 Detailed labeled precision on the test set

Table 4 presented the labeled precision of each type of semantic role. It demonstrates that with respect to ARGMs and ARGXs, the hierarchical system outperforms the baseline system. Furthermore, the improvement on ARGXs is greater than that of ARGMs. All types of numbered arguments get improvement in the hierarchical architecture, especially ARG1, ARG4 and ARG3. Although the performances of some types of the ARGMs decreased, the performances of all types of the ARGMs which occurs more than 100 times increased, including ADV (adverbials), LOC (locatives), MNR (manner markers) and TMP (temporal markers).

After the hierarchical system was built, we tried to integrate the idea of exploiting argument interdependence into our system. We extract the semantic context features in a linear order, with the window size from [0,0] to [-3,3]. Larger window sizes are of little value since too few arguments have more than 6 other arguments in context. The results are presented in table 5.

	Baseline	Hierarchical
Base	93.38%	94.31%
+window selection	93.38%	**94.68%**

Table 5 integrating window selection into our system

"Base" stands for the hierarchical system built above, without semantic context features. "+window selection" indicates the new system which has utilized the idea of exploiting argument interdependence. The best window sizes for the baseline system, ARGX and ARGM processes in the hierarchical system are [0,0], [-1,1], [0,0] respectively, which were determined by testing on the development set. We can find that only for the ARGX process, the semantic context features are useful. For the baseline system and the ARGM process, exploiting argument interdependence does not help improve the performance. This conclusion is different from Jiang et al. (2004), but it can be explained in the following way.

In fact, the interdependence only exists among core arguments. For ARGMs, it is a different thing. An ARGM cannot provide any information about the type of the arguments close to it and the semantic context features does not help the classification of ARGMs. Also, take the sentence in section 2 as an example, the fact that 截止目前 (until now) is ARGM-TMP cannot raise the probability that 保险公司 (the insurance company) is ARG0 or 已 (has) is ARGM-ADV and vice versa. However, if we know that 保险公司 (the insurance company) is ARG0, at least the phrase 保险服务 (insurance services) can never be ARG0. The semantic context features extracted from or for ARGMs will do

[4] From the name, TBERR possibly indicates the labeled errors in Chinese PropBank. However, we did not find any explanations, so we just put it here and group it to ARGM.

harm to the improvement of the system, since they are irrelative information. Because of the same reason, the performance of base system also decreased when semantic context features were extracted, since the core arguments and the ARGMs are mixed together in the baseline system.

But for the ARGX sub task of our hierarchical system, since we have separated the numbered arguments and ARGMs first, the influences of ARGMs can be eliminated. This made the interdependence of core arguments can be directly explored from the extraction of semantic context features. So the ARGX sub task is improved.

To prove that our method is effective, we also make a comparison between the performances of our system and Xue and Palmer (2005), Xue (2008). Xue (2008) is the best SRL system until now and it has the same data setting with ours. The results are presented in Table 6.

X & P (2005)	Xue(2008)	Ours
93.9%	94.1%	**94.68%**

Table 6. Comparison with previous systems

We have to point out that all the three systems are based on Gold standard parsing. From the table 6, we can find that our system is better than both of the related systems. Our system has outperformed Xue (2008) with a relative error reduction rate of 9.8%.

7 Conclusions and Future Work

In this paper, we have divided all the semantic roles into two groups according to their semantic relations with the verb. After the grouping of the semantic roles was made, we designed a hierarchical semantic role classifier. To capture the accurate information of different semantic role groups, we designed a simple feature selection algorithm to calibrate features for each sub task of SRC. It was very encouraging that the hierarchical SRC system outperformed the strong baseline built with traditional methods. And the selected features could be explained, which in turn proves that the linguistic discrepancy of semantic role groups not only exists but also can be captured. Then we integrated the idea of exploiting argument interdependence to further improve the performance of our system and explained linguistically why the results of our system were different from the ones in previous research.

Although we make discriminations of arguments and adjuncts, the analysis is still coarse-grained. Yi et al. (2007) has made the first attempt working on the single semantic role level to make further improvement. However, the impact of this idea is limited due to that the amount of the research target, ARG2, is few in PropBank. What if we could extend the idea of hierarchical architecture to the single semantic role level? Would that help the improvement of SRC?

Acknowledgements

This work was supported by National Natural Science Foundation of China under Grant No. 60303003 and National Social Science Foundation of China under Grant No. 06BYY048.

We want to thank Nianwen Xue, for his generous help at the beginning of this work. And thanks to the anonymous reviewers, for their valuable comments on this paper.

References

Baker, Collin F., Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of the 17th international conference on Computational linguistics*, Montreal, Canada.

Boas, Hans C. 2002. Bilingual FrameNet dictionaries for machine translation. In *Proceedings of LREC 2002*, Las Palmas, Spain.

Carreras, Xavier and Lluís Màrquez. 2004. Introduction to the conll-2004 shared task: Semantic role labeling. In *Proceedings of the Eighth Conference on Natural Language Learning,* Boston, Massachusetts.

Carreras, Xavier and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the Nineth Conference on Natural Language Learning,* Ann Arbor, Michigan.

Màrquez, Lluís, Xavier Carreras, Kenneth C. Litkowski, Suzanne Stevenson. 2008. Semantic Role Labeling: An Introduction to the Special Issue, *Computational linguistics.* 34(2):146-159..

Gildea, Daniel and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3): 245-288.

Jiang, Zheng Ping, Jia Li, Hwee Tou Ng. 2005. Semantic Argument Classification Exploiting Argument Interdependence. In *19th International Joint Conference on Artificial Intelligence.* Edinburgh, Scotland.

Kingsbury, Paul and Martha Palmer. 2002. From TreeBank to PropBank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, Las Palmas, Spain.

Kipper, Karin, Hoa Trang Dang, and Martha Palmer. 2000. Class-based construction of a verb lexicon. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, Austin, Texas, USA.

Moschitti. Alessandro. 2004. A Study on Convolution Kernels for Shallow Statistic Parsing. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, Barcelona, Spain.

Moschitti, Alessandro, Ana-Maria Giuglea, Bonaventura Coppola, and Roberto Basili. 2005. *Hierarchical semantic role labeling*. In *Proceedings of the Nineth Conference on Natural Language Learning,* Ann Arbor, Michigan.

Narayanan, Srini and Sanda Harabagiu. 2004. Question answering based on semantic structures. In *Proceedings of the 20th International Conference on Computational Linguistics*, Geneva, Switzerland.

Pradhan, Sameer, Kadri Hacioglu, Valerie Kruglery, Wayne Ward, James H. Martin, Daniel Jurafskyz. 2004. Support vector learning for semantic argument classification. *Machine Learning Journal*, 60(1-3):11-39.

Sun, Honglin and Daniel Jurafsky. 2004. Shallow Semantic Parsing of Chinese. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, Boston, Massachusetts.

Surdeanu, Mihai, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, Ann Arbor, Michigan.

Wang, Hui, Weidong Zhan, Shiwen Yu. 2003. The Specification of The Semantic Knowledge-base of Contemporary Chinese, In *Journal of Chinese Language and Computing*, 13(2):159-176.

Xue, Nianwen and Martha Palmer. 2003. Annotating the Propositions in the Penn Chinese Treebank. In *Proceedings of the 2nd SIGHAN Workshop on Chinese Language Processing*, Sapporo, Japan.

Xue, Nianwen and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Barcelona, Spain.

Xue, Nianwen and Martha Palmer. 2005. Automatic semantic role labeling for Chinese verbs. In *19th International Joint Conference on Artificial Intelligence.* Edinburgh, Scotland.

Xue, Nianwen, Fei Xia, Fu dong Chiou, and Martha Palmer. 2005. The Penn Chinese TreeBank: Phrase Structure Annotation of a Large Corpus. Natural Language Engineering, 11(2):207–238.

Xue, Nianwen. 2008. Labeling Chinese predicates with semantic roles. *Computational linguistics.* 34(2):225-255.

Yi, Szu-ting, Edward Loper, Martha Palmer. 2007. Can Semantic Roles Generalize Across Genres? In *Proceedings of Human Language Technologies and The Conference of the North American Chapter of the Association for Computational Linguistics*, Rochester, NY, USA.

Zhang, Min, Wanxiang Che, Ai Ti Aw, Chew Lim Tan, Guodong Zhou, Ting Liu, Sheng Li. 2007. A Grammar-driven Convolution Tree Kernel for Semantic Role Classification, in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, Prague, Czech Republic.

Bayesian Unsupervised Topic Segmentation

Jacob Eisenstein and **Regina Barzilay**
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
77 Massachusetts Ave., Cambridge MA 02139
{jacobe,regina}@csail.mit.edu

Abstract

This paper describes a novel Bayesian approach to unsupervised topic segmentation. Unsupervised systems for this task are driven by *lexical cohesion*: the tendency of well-formed segments to induce a compact and consistent lexical distribution. We show that lexical cohesion can be placed in a Bayesian context by modeling the words in each topic segment as draws from a multinomial language model associated with the segment; maximizing the observation likelihood in such a model yields a lexically-cohesive segmentation. This contrasts with previous approaches, which relied on hand-crafted cohesion metrics. The Bayesian framework provides a principled way to incorporate additional features such as cue phrases, a powerful indicator of discourse structure that has not been previously used in unsupervised segmentation systems. Our model yields consistent improvements over an array of state-of-the-art systems on both text and speech datasets. We also show that both an entropy-based analysis and a well-known previous technique can be derived as special cases of the Bayesian framework.[1]

1 Introduction

Topic segmentation is one of the fundamental problems in discourse analysis, where the task is to divide a text into a linear sequence of topically-coherent segments. Hearst's TEXTTILING (1994) introduced the idea that unsupervised segmentation can be driven by *lexical cohesion*, as high-quality segmentations feature homogeneous lexical distributions within each topic segment. Lexical cohesion has provided the inspiration for several successful systems (e.g., Utiyama and Isahara, 2001; Galley et al.2003; Malioutov and Barzilay, 2006), and is currently the dominant approach to unsupervised topic segmentation.

But despite the effectiveness of lexical cohesion for unsupervised topic segmentation, it is clear that there are other important indicators that are ignored by the current generation of unsupervised systems. For example, consider *cue phrases*, which are explicit discourse markers such as "now" or "however" (Grosz and Sidner, 1986; Hirschberg and Litman, 1993; Knott, 1996). Cue phrases have been shown to be a useful feature for supervised topic segmentation (Passonneau and Litman, 1993; Galley et al., 2003), but cannot be incorporated by current unsupervised models. One reason for this is that existing unsupervised methods use arbitrary, hand-crafted metrics for quantifying lexical cohesion, such as weighted cosine similarity (Hearst, 1994; Malioutov and Barzilay, 2006). Without supervision, it is not possible to combine such metrics with additional sources of information. Moreover, such hand-crafted metrics may not generalize well across multiple datasets, and often include parameters which must be tuned on development sets (Malioutov and Barzilay, 2006; Galley et al., 2003).

In this paper, we situate lexical cohesion in a Bayesian framework, allowing other sources of information to be incorporated without the need for labeled data. We formalize lexical cohesion in a generative model in which the text for each seg-

[1]Code and materials for this work are available at `http://groups.csail.mit.edu/rbg/code/bayesseg/`.

ment is produced by a distinct lexical distribution. Lexically-consistent segments are favored by this model because probability mass is conserved for a narrow subset of words. Thus, lexical cohesion arises naturally through the generative process, and other sources of information – such as cue words – can easily be incorporated as emissions from the segment boundaries.

More formally, we treat the words in each sentence as draws from a language model associated with the topic segment. This is related to topic-modeling methods such as latent Dirichlet allocation (LDA; Blei et al. 2003), but here the induced topics are tied to a linear discourse structure. This property enables a dynamic programming solution to find the exact maximum-likelihood segmentation. We consider two approaches to handling the language models: estimating them explicitly, and integrating them out, using the Dirichlet Compound Multinomial distribution (also known as the multivariate Polya distribution).

We model cue phrases as generated from a separate multinomial that is shared across all topics and documents in the dataset; a high-likelihood model will obtain a compact set of cue phrases. The addition of cue phrases renders our dynamic programming-based inference inapplicable, so we design a sampling-based inference technique. This algorithm can learn in a completely unsupervised fashion, but it also provides a principled mechanism to improve search through the addition of declarative linguistic knowledge. This is achieved by biasing the selection of samples towards boundaries with known cue phrases; this does not change the underlying probabilistic model, but guides search in the direction of linguistically-plausible segmentations.

We evaluate our algorithm on corpora of spoken and written language, including the benchmark ICSI meeting dataset (Janin et al., 2003) and a new textual corpus constructed from the contents of a medical textbook. In both cases our model achieves performance surpassing multiple state-of-the-art baselines. Moreover, we demonstrate that the addition of cue phrases can further improve segmentation performance over cohesion-based methods.

In addition to the practical advantages demonstrated by these experimental results, our model reveals interesting theoretical properties. Other researchers have observed relationships between discourse structure and entropy (e.g., Genzel and Charniak, 2002). We show that in a special case of our model, the segmentation objective is equal to a weighted sum of the negative entropies for each topic segment. This finding demonstrates that a relationship between discourse segmentation and entropy is a natural consequence of modeling topic structure in a generative Bayesian framework. In addition, we show that the benchmark segmentation system of Utiyama and Isahara (2001) can be viewed as another special case of our Bayesian model.

2 Related Work

Existing unsupervised cohesion-based approaches can be characterized in terms of the metric used to quantify cohesion and the search technique. Galley et al. (2003) characterize cohesion in terms of lexical chains – repetitions of a given lexical item over some fixed-length window of sentences. In their unsupervised model, inference is performed by selecting segmentation points at the local maxima of the cohesion function. Malioutov and Barzilay (2006) optimize a normalized minimum-cut criteria based on a variation of the cosine similarity between sentences. Most similar to our work is the approach of Utiyama and Isahara (2001), who search for segmentations with compact language models; as shown in Section 3.1.1, this can be viewed as a special case of our model. Both of these last two systems use dynamic programming to search the space of segmentations.

An alternative Bayesian approach to segmentation was proposed by Purver et al. (2006). They assume a set of documents that is characterized by some number of hidden topics that are shared across multiple documents. They then build a linear segmentation by adding a switching variable to indicate whether the topic distribution for each sentence is identical to that of its predecessor. Unlike Purver et al., we do not assume a dataset in which topics are shared across multiple documents; indeed, our model can be applied to single documents individually. Additionally, the inference procedure of Purver et al. requires sampling multiple layers of hidden variables. In contrast, our inference procedure leverages the nature of linear segmentation to search only in the space of segmentation points.

The relationship between discourse structure and cue phrases has been studied extensively; for an early example of computational work on this topic, see (Grosz, 1977). Passonneau and Litman (1993) were the first to investigate the relationship between cue phrases and linear segmentation. More recently, cue phrases have been applied to topic segmentation in the supervised setting. In a supervised system that is distinct from the unsupervised model described above, Galley et al. (2003) automatically identify candidate cue phrases by mining labeled data for words that are especially likely to appear at segment boundaries; the presence of cue phrases is then used as a feature in a rule-based classifier for linear topic segmentation. Elsner and Charniak (2008) specify a list of cue phrases by hand; the cue phrases are used as a feature in a maximum-entropy classifier for conversation disentanglement. Unlike these approaches, we identify candidate cue phrases automatically from unlabeled data and incorporate them in the topic segmentation task without supervision.

3 Lexical Cohesion in a Bayesian Framework

The core idea of lexical cohesion is that topically-coherent segments demonstrate compact and consistent lexical distributions (Halliday and Hasan, 1976). Lexical cohesion can be placed in a probabilistic context by modeling the words in each topic segment as draws from a multinomial language model associated with the segment. Formally, if sentence t is in segment j, then the bag of words $\mathbf{x}_t$ is drawn from the multinomial language model θ_j. This is similar in spirit to hidden topic models such as latent Dirichlet allocation (Blei et al., 2003), but rather than assigning a hidden topic to each word, we constrain the topics to yield a linear segmentation of the document.

We will assume that topic breaks occur at sentence boundaries, and write z_t to indicate the topic assignment for sentence t. The observation likelihood is,

$$p(\mathbf{X}|\mathbf{z},\Theta) = \prod_{t}^{T} p(\mathbf{x}_t|\theta_{z_t}), \tag{1}$$

where $\mathbf{X}$ is the set of all T sentences, $\mathbf{z}$ is the vector of segment assignments for each sentence, and Θ is the set of all K language models.[2] A linear segmentation is ensured by the additional constraint that z_t must be equal to either z_{t-1} (the previous sentence's segment) or $z_{t-1}+1$ (the next segment).

To obtain a high likelihood, the language models associated with each segment should concentrate their probability mass on a compact subset of words. Language models that spread their probability mass over a broad set of words will induce a lower likelihood. This is consistent with the principle of lexical cohesion.

Thus far, we have described a segmentation in terms of two parameters: the segment indices $\mathbf{z}$, and the set of language models Θ. For the task of segmenting documents, we are interested only in the segment indices, and would prefer not to have to search in the space of language models as well. We consider two alternatives: taking point estimates of the language models (Section 3.1), and analytically marginalizing them out (Section 3.2).

3.1 Setting the language model to the posterior expectation

One way to handle the language models is to choose a single point estimate for each set of segmentation points $\mathbf{z}$. Suppose that each language model is drawn from a symmetric Dirichlet prior: $\theta_j \sim \text{Dir}(\theta_0)$. Let $\mathbf{n}_j$ be a vector in which each element is the sum of the lexical counts over all the sentences in segment j: $n_{j,i} = \sum_{\{t:z_t=j\}} m_{t,i}$, where $m_{t,i}$ is the count of word i in sentence t. Assuming that each $\mathbf{x}_t \sim \theta_j$, then the posterior distribution for θ_j is Dirichlet with vector parameter $\mathbf{n}_j + \theta_0$ (Bernardo and Smith, 2000). The expected value of this distribution is the multinomial distribution $\hat{\theta}_j$, where,

$$\hat{\theta}_{j,i} = \frac{n_{j,i} + \theta_0}{\sum_i^W n_{j,i} + W\theta_0}. \tag{2}$$

In this equation, W indicates the number of words in the vocabulary. Having obtained an estimate for the language model $\hat{\theta}_j$, the observed data likelihood for segment j is a product over each sentence in the segment,

[2]Our experiments will assume that the number of topics K is known. This is common practice for this task, as the desired number of segments may be determined by the user (Malioutov and Barzilay, 2006).

$$p(\{\mathbf{x}_t : z_t = j\}|\hat{\theta}_j) = \prod_{\{t:z_t=j\}} \prod_{i \in \mathbf{x}_t} \hat{\theta}_{j,i} \quad (3)$$

$$= \prod_{\{t:z_t=j\}} \prod_{i}^{W} \hat{\theta}_{j,i}^{m_{t,i}} \quad (4)$$

$$= \prod_{i}^{W} \hat{\theta}_{j,i}^{n_{j,i}}. \quad (5)$$

By viewing the likelihood as a product over all terms in the vocabulary, we observe interesting connections with prior work on segmentation and information theory.

3.1.1 Connection to previous work

In this section, we explain how our model generalizes the well-known method of Utiyama and Isahara (2001; hereafter U&I). As in our work, Utiyama and Isahara propose a probabilistic framework based on maximizing the compactness of the language models induced for each segment. Their likelihood equation is identical to our equations 3-5. They then define the language models for each segment as $\hat{\theta}_{j,i} = \frac{n_{j,i}+1}{W+\sum_i^W n_{j,i}}$, without rigorous justification. This form is equivalent to Laplacian smoothing (Manning and Schütze, 1999), and is a special case of our equation 2, with $\theta_0 = 1$. Thus, the language models in U&I can be viewed as the expectation of the posterior distribution $p(\theta_j|\{\mathbf{x}_t : z_t = j\}, \theta_0)$, in the special case that $\theta_0 = 1$. Our approach generalizes U&I and provides a Bayesian justification for the language models that they apply. The remainder of the paper further extends this work by marginalizing out the language model, and by adding cue phrases. We empirically demonstrate that these extensions substantially improve performance.

3.1.2 Connection to entropy

Our model also has a connection to entropy, and situates entropy-based segmentation within a Bayesian framework. Equation 1 defines the objective function as a product across sentences; using equations 3-5 we can decompose this across segments instead. Working in logarithms,

$$\log p(\mathbf{X}|\mathbf{z}, \hat{\Theta}) = \sum_{t}^{T} \log p(\mathbf{x}_t|\hat{\theta}_{z_t})$$

$$= \sum_{j}^{K} \sum_{\{t:z_t=j\}} \log p(\mathbf{x}_t|\hat{\theta}_j)$$

$$= \sum_{j}^{K} \sum_{i}^{W} n_{j,i} \log \hat{\theta}_{j,i} \quad (6)$$

The last line substitutes in the logarithm of equation 5. Setting $\theta_0 = 0$ and rearranging equation 2, we obtain $n_{j,i} = N_j \hat{\theta}_{j,i}$, with $N_j = \sum_i^W n_{j,i}$, the total number of words in segment j. Substituting this into equation 6, we obtain

$$\log p(\mathbf{X}|\mathbf{z}, \hat{\Theta}) = \sum_{j}^{K} N_j \sum_{i} \hat{\theta}_{j,i} \log \hat{\theta}_{j,i}$$

$$= \sum_{j}^{K} N_j H(\hat{\theta}_j),$$

where $H(\hat{\theta}_j)$ is the negative entropy of the multinomial $\hat{\theta}_j$. Thus, with $\theta_0 = 0$, the log conditional probability in equation 6 is optimized by a segmentation that minimizes the weighted sum of entropies per segment, where the weights are equal to the segment lengths. This result suggests intriguing connections with prior work on the relationship between entropy and discourse structure (e.g., Genzel and Charniak, 2002; Sporleder and Lapata, 2006).

3.2 Marginalizing the language model

The previous subsection uses point estimates of the language models to reveal connections to entropy and prior work on segmentation. However, point estimates are theoretically unsatisfying from a Bayesian perspective, and better performance may be obtained by marginalizing over all possible language models:

$$
\begin{aligned}
p(\mathbf{X}|\mathbf{z},\theta_0) &= \prod_j^K \prod_{\{t:z_t=j\}} p(\mathbf{x}_t|\theta_0) \\
&= \prod_j^K \int d\theta_j \prod_{\{t:z_t=j\}} p(x_t|\theta_j)p(\theta_j|\theta_0) \\
&= \prod_j^K p_{dcm}(\{\mathbf{x}_t : z_t = j\}|\theta_0), \qquad (7)
\end{aligned}
$$

where p_{dcm} refers to the Dirichlet compound multinomial distribution (DCM), also known as the multivariate Polya distribution (Johnson et al., 1997). The DCM distribution expresses the expectation over all multinomial language models, when conditioning on the Dirichlet prior θ_0. When θ_0 is a symmetric Dirichlet prior,

$$
\begin{aligned}
&p_{dcm}(\{\mathbf{x}_t : z_t = j\}|\theta_0) \\
&\quad = \frac{\Gamma(W\theta_0)}{\Gamma(N_j + W\theta_0)} \prod_i^W \frac{\Gamma(n_{j,i} + W\theta_0)}{\Gamma(\theta_0)},
\end{aligned}
$$

where $n_{j,i}$ is the count of word i in segment j, and $N_j = \sum_i^W n_{j,i}$, the total number of words in the segment. The symbol Γ refers to the Gamma function, an extension of the factorial function to real numbers. Using the DCM distribution, we can compute the data likelihood for each segment from the lexical counts over the entire segment. The overall observation likelihood is a product across the likelihoods for each segment.

3.3 Objective function and inference

The optimal segmentation maximizes the joint probability,

$$p(\mathbf{X},\mathbf{z}|\theta_0) = p(\mathbf{X}|\mathbf{z},\theta_0)p(\mathbf{z}).$$

We assume that $p(\mathbf{z})$ is a uniform distribution over valid segmentations, and assigns no probability mass to invalid segmentations. The data likelihood is defined for point estimate language models in equation 5 and for marginalized language models in equation 7. Note that equation 7 is written as a product over segments. The point estimates for the language models depend only on the counts within each segment, so the overall likelihood for the point-estimate version also decomposes across segments.

Any objective function that can be decomposed into a product across segments can be maximized using dynamic programming. We define $B(t)$ as the value of the objective function for the optimal segmentation up to sentence t. The contribution to the objective function from a single segment between sentences t' and t is written,

$$b(t',t) = p(\{\mathbf{x}_{t'} \ldots \mathbf{x}_t\}|\mathbf{z}_{t' \ldots t} = j)$$

The maximum value of the objective function is then given by the recurrence relation, $B(t) = \max_{t'<t} B(t')b(t'+1,t)$, with the base case $B(0) = 1$. These values can be stored in a table of size T (equal to the number of sentences); this admits a dynamic program that performs inference in polynomial time.[3] If the number of segments is specified in advance, the dynamic program is slightly more complex, with a table of size TK.

3.4 Priors

The Dirichlet compound multinomial integrates over language models, but we must still set the prior θ_0. We can re-estimate this prior based on the observed data by interleaving gradient-based search in a Viterbi expectation-maximization framework (Gauvain and Lee, 1994). In the E-step, we estimate a segmentation $\hat{\mathbf{z}}$ of the dataset, as described in Section 3.3. In the M-step, we maximize $p(\theta_0|\mathbf{X},\hat{\mathbf{z}}) \propto p(\mathbf{X}|\theta_0,\hat{\mathbf{z}})p(\theta_0)$. Assuming a non-informative hyperprior $p(\theta_0)$, we maximize the likelihood in Equation 7 across all documents. The maximization is performed using a gradient-based search; the gradients are dervied by Minka (2003). This procedure is iterated until convergence or a maximum of twenty iterations.

4 Cue Phrases

One of the key advantages of a Bayesian framework for topic segmentation is that it permits the principled combination of multiple data sources, even

[3]This assumes that the objective function for individual segments can also be computed efficiently. In our case, we need only keep vectors of counts for each segment, and evaluate probability density functions over the counts.

without labeled data. We are especially interested in cue phrases, which are explicit markers for discourse structure, such as "now" or "first" (Grosz and Sidner, 1986; Hirschberg and Litman, 1993; Knott, 1996). Cue phrases have previously been used in supervised topic segmentation (e.g., Galley et al. 2003); we show how they can be used in an unsupervised setting.

The previous section modeled lexical cohesion by treating the bag of words in each sentence as a series of draws from a multinomial language model indexed by the topic segment. To incorporate cue phrases, this generative model is modified to reflect the idea that some of the text will be topic-specific, but other terms will be topic-neutral cue phrases that express discourse structure. This idea is implemented by drawing the text at each topic boundary from a special language model ϕ, which is shared across all topics and all documents in the dataset.

For sentences that are not at segment boundaries, the likelihood is as before: $p(\mathbf{x}_t|\mathbf{z}, \Theta, \phi) = \prod_{i \in \mathbf{x}_t} \theta_{z_t,i}$. For sentences that immediately follow segment boundaries, we draw the first ℓ words from ϕ instead. Writing $\mathbf{x}_t^{(\ell)}$ for the ℓ cue words in $\mathbf{x}_t$, and $\tilde{\mathbf{x}}_t$ for the remaining words, the likelihood for a segment-initial sentence is,

$$p(\mathbf{x}_t|z_t \neq z_{t-1}, \Theta, \phi) = \prod_{i \in \mathbf{x}_t^{(\ell)}} \phi_i \prod_{i \in \tilde{\mathbf{x}}_t} \theta_{z_t,i}.$$

We draw ϕ from a symmetric Dirichlet prior ϕ_0. Following prior work (Galley et al., 2003; Litman and Passonneau, 1995), we consider only the first word of each sentence as a potential cue phrase; thus, we set $\ell = 1$ in all experiments.

4.1 Inference

To estimate or marginalize the language models Θ and ϕ, it is necessary to maintain lexical counts for each segment and for the segment boundaries. The counts for ϕ are summed across every segment in the entire dataset, so shifting a boundary will affect the probability of every segment, not only the adjacent segments as before. Thus, the factorization that enabled dynamic programming inference in Section 3.3 is no longer applicable. Instead, we must resort to approximate inference.

Sampling-based inference is frequently used in related Bayesian models. Such approaches build a stationary Markov chain by repeatedly sampling among the hidden variables in the model. The most commonly-used sampling-based technique is Gibbs sampling, which iteratively samples from the conditional distribution of each hidden variable (Bishop, 2006). However, Gibbs sampling is slow to converge to a stationary distribution when the hidden variables are tightly coupled. This is the case in linear topic segmentation, due to the constraint that $z_t \in \{z_{t-1}, z_{t-1}+1\}$ (see Section 3).

For this reason, we apply the more general Metropolis-Hastings algorithm, which permits sampling arbitrary transformations of the latent variables. In our framework, such transformations correspond to moves through the space of possible segmentations. A new segmentation $\mathbf{z}'$ is drawn from the previous hypothesized segmentation $\mathbf{z}$ based on a *proposal distribution* $q(\mathbf{z}'|\mathbf{z})$.[4] The probability of accepting a proposed transformation depends on the ratio of the joint probabilities and a correction term for asymmetries in the proposal distribution:

$$p_{\text{accept}}(\mathbf{z} \rightarrow \mathbf{z}') = \min\left\{1, \frac{p(\mathbf{X}, \mathbf{z}'|\theta_0, \phi_0)}{p(\mathbf{X}, \mathbf{z}|\theta_0, \phi_0)} \frac{q(\mathbf{z}|\mathbf{z}')}{q(\mathbf{z}'|\mathbf{z})}\right\}.$$

The Metropolis-Hastings algorithm guarantees that by accepting samples at this ratio, our sampling procedure will converge to the stationary distribution for the hidden variables $\mathbf{z}$. When cue phrases are included, the observation likelihood is written:

$$p(\mathbf{X}|\mathbf{z}, \Theta, \phi) = \prod_{\{t: z_t \neq z_{t-1}\}} \prod_{i \in \mathbf{x}_t^{(\ell)}} \phi_i \prod_{i \in \tilde{\mathbf{x}}_t} \theta_{z_t,i} \times \prod_{\{t: z_t = z_{t-1}\}} \prod_{i \in \mathbf{x}_t} \theta_{z_t,i}.$$

As in Section 3.2, we can marginalize over the language models. We obtain a product of DCM distributions: one for each segment, and one for all cue phrases in the dataset.

4.2 Proposal distribution

Metropolis-Hastings requires a proposal distribution to sample new configurations. The proposal distri-

[4]Because the cue phrase language model ϕ is used across the entire dataset, transformations affect the likelihood of all documents in the corpus. For clarity, our exposition will focus on the single-document case.

bution does not affect the underlying probabilistic model – Metropolis-Hastings will converge to the same underlying distribution for any non-degenerate proposal. However, a well-chosen proposal distribution can substantially speed convergence.

Our basic proposal distribution selects an existing segmentation point with uniform probability, and considers a set of local moves. The proposal is constructed so that no probability mass is allocated to moves that change the order of segment boundaries, or merge two segments; one consequence of this restriction is that moves cannot add or remove segments.[5] We set the proposal distribution to decrease exponentially with the move distance, thus favoring incremental transformations to the segmentation.

More formally, let $d(\mathbf{z} \rightarrow \mathbf{z}') > 0$ equal the distance that the selected segmentation point is moved when we transform the segmentation from $\mathbf{z}$ to $\mathbf{z}'$. We can write the proposal distribution $q(\mathbf{z}' \mid \mathbf{z}) \propto c(\mathbf{z} \rightarrow \mathbf{z}')d(\mathbf{z} \rightarrow \mathbf{z}')^{\lambda}$, where $\lambda < 0$ sets the rate of exponential decay and c is an indicator function enforcing the constraint that the moves do not reach or cross existing segmentation points.[6]

We can also incorporate declarative linguistic knowledge by biasing the proposal distribution in favor of moves that place boundaries near known cue phrase markers. We multiply the unnormalized chance of proposing a move to location $\mathbf{z} \rightarrow \mathbf{z}'$ by a term equal to one plus the number of candidate cue phrases in the segment-initial sentences in the new configuration $\mathbf{z}'$, written $\textsf{num-cue}(\mathbf{z}')$. Formally, $q_{\text{ling}}(\mathbf{z}' \mid \mathbf{z}') \propto (1 + \textsf{num-cue}(\mathbf{z}'))q(\mathbf{z}' \mid \mathbf{z})$. We use a list of cue phrases identified by Hirschberg and Litman (1993). We evaluate our model with both the basic and linguistically-enhanced proposal distributions.

4.3 Priors

As in section 3.4, we set the priors θ_0 and ϕ_0 using gradient-based search. In this case, we perform gradient-based optimization after epochs of 1000 Metropolis-Hasting steps. Interleaving sampling-based inference with direct optimization of parameters can be considered a form of Monte Carlo Expectation-Maximization (MCEM; Wei and Tanner, 1990).

5 Experimental Setup

Corpora We evaluate our approach on corpora from two different domains: transcribed meetings and written text.

For multi-speaker meetings, we use the ICSI corpus of meeting transcripts (Janin et al., 2003), which is becoming a standard for speech segmentation (e.g., Galley et al. 2003; Purver et al. 2006). This dataset includes transcripts of 75 multi-party meetings, of which 25 are annotated for segment boundaries.

For text, we introduce a dataset in which each document is a chapter selected from a medical textbook (Walker et al., 1990).[7] The task is to divide each chapter into the sections indicated by the author. This dataset contains 227 chapters, with 1136 sections (an average of 5.00 per chapter). Each chapter contains an average of 140 sentences, giving an average of 28 sentences per segment.

Metrics All experiments are evaluated in terms of the commonly-used P_k (Beeferman et al., 1999) and WindowDiff (WD) (Pevzner and Hearst, 2002) scores. Both metrics pass a window through the document, and assess whether the sentences on the edges of the window are properly segmented with respect to each other. WindowDiff is stricter in that it requires that the number of intervening segments between the two sentences be identical in the hypothesized and the reference segmentations, while P_k only asks whether the two sentences are in the same segment or not. P_k and WindowDiff are penalties, so lower values indicate better segmentations. We use the evaluation source code provided by Malioutov and Barzilay (2006).

System configuration We evaluate our Bayesian approach both with and without cue phrases. Without cue phrases, we use the dynamic programming inference described in section 3.3. This system is referred to as BAYESSEG in Table 1. When adding

[5]Permitting moves to change the number of segments would substantially complicate inference.

[6]We set $\lambda = -\frac{1}{\textsf{max-move}}$, where $\textsf{max-move}$ is the maximum move-length, set to 5 in our experiments. These parameters affect the rate of convergence but are unrelated to the underlying probability model. In the limit of enough samples, all non-pathological settings will yield the same segmentation results.

[7]The full text of this book is available for free download at http://onlinebooks.library.upenn.edu.

cue phrases, we use the Metropolis-Hastings model described in 4.1. Both basic and linguistically-motivated proposal distributions are evaluated (see Section 4.2); these are referred to as BAYESSEG-CUE and BAYESSEG-CUE-PROP in the table.

For the sampling-based systems, results are averaged over five runs. The initial configuration is obtained from the dynamic programming inference, and then 100,000 sampling iterations are performed. The final segmentation is obtained by annealing the last 25,000 iterations to a temperature of zero. The use of annealing to obtain a maximum *a posteriori* (MAP) configuration from sampling-based inference is common (e.g., Finkel 2005; Goldwater 2007). The total running time of our system is on the order of three minutes per document. Due to memory constraints, we divide the textbook dataset into ten parts, and perform inference in each part separately. We may achieve better results by performing inference over the entire dataset simultaneously, due to pooling counts for cue phrases across all documents.

Baselines We compare against three competitive alternative systems from the literature: U&I (Utiyama and Isahara, 2001); LCSEG (Galley et al., 2003); MCS (Malioutov and Barzilay, 2006). All three systems are described in the related work (Section 2). In all cases, we use the publicly available executables provided by the authors.

Parameter settings For LCSEG, we use the parameter values specified in the paper (Galley et al., 2003). MCS requires parameter settings to be tuned on a development set. Our corpora do not include development sets, so tuning was performed using the lecture transcript corpus described by Malioutov and Barzilay (2006). Our system does not require parameter tuning; priors are re-estimated as described in Sections 3.4 and 4.3. U&I requires no parameter tuning, and is used "out of the box." In all experiments, we assume that the number of desired segments is provided.

Preprocessing Standard preprocessing techniques are applied to the text for all comparisons. The Porter (1980) stemming algorithm is applied to group equivalent lexical items. A set of stop-words is also removed, using the same list originally employed by several competitive systems (Choi, 2000; Utiyama and Isahara, 2001; Malioutov and Barzilay, 2006).

Textbook	P_k	WD
U&I	.370	.376
MCS	.368	.382
LCSEG	.370	.385
BAYESSEG	**.339**	**.353**
BAYESSEG-CUE	**.339**	**.353**
BAYESSEG-CUE-PROP	.343	.355
Meetings	P_k	WD
U&I	.297	.347
MCS	.370	.411
LCSEG	.309	.322
BAYESSEG	.264	.319
BAYESSEG-CUE	.261	.316
BAYESSEG-CUE-PROP	**.258**	**.312**

Table 1: Comparison of segmentation algorithms. Both metrics are penalties, so lower scores indicate better performance. BAYESSEG is the cohesion-only Bayesian system with marginalized language models. BAYESSEG-CUE is the Bayesian system with cue phrases. BAYESSEG-CUE-PROP adds the linguistically-motivated proposal distribution.

6 Results

Table 1 presents the performance results for three instantiations of our Bayesian framework and three competitive alternative systems. As shown in the table, the Bayesian models achieve the best results on both metrics for both corpora. On the medical textbook corpus, the Bayesian systems achieve a raw performance gain of 2-3% with respect to all baselines on both metrics. On the ICSI meeting corpus, the Bayesian systems perform 4-5% better than the best baseline on the P_k metric, and achieve smaller improvement on the WindowDiff metric. The results on the meeting corpus also compare favorably with the topic-modeling method of Purver et al. (2006), who report a P_k of .289 and a WindowDiff of .329.

Another observation from Table 1 is that the contribution of cue phrases depends on the dataset. Cue phrases improve performance on the meeting corpus, but not on the textbook corpus. The effectiveness of cue phrases as a feature depends on whether the writer or speaker uses them consistently. At the

Meetings		Textbook	
okay*	234.4	**the**	1345.9
I	212.6	**this**	14.3
so*	113.4	it	4.1
um	91.7	these	4.1
and*	67.3	a	2.9
yeah	10.5	on	2.1
but*	9.4	most	2.0
uh	4.8	heart	1.8
right	2.4	creating	1.8
agenda	1.3	hundred	1.8

Table 2: Cue phrases selected by our unsupervised model, sorted by chi-squared. Boldface indicates that the chi-squared value is significant at the level of $p < .01$. Asterisks indicate cue phrases that were extracted by the supervised procedure of Galley et al. (2003).

same time, the addition of cue phrases prevents the use of exact inference techniques, which may explain the decline in results for the meetings dataset.

To investigate the quality of the cue phrases that our model extracts, we list its top ten cue phrases for each dataset in Table 2. Cue phrases are ranked by their chi-squared value, which is computed based on the number of occurrences for each word at the beginning of a hypothesized segment, as compared to the expectation. For cue phrases listed in bold, the chi-squared value is statistically significant at the level of $p < .01$, indicating that the frequency with which the cue phrase appears at the beginning of segments is unlikely to be a chance phenomenon.

As shown in the left column of the table, our model has identified several strong cue phrases from the meeting dataset which appear to be linguistically plausible. Galley et al. (2003) performed a similar chi-squared analysis, but used the true segment boundaries in the labeled data; this can be thought of as a sort of ground truth. Four of the ten cue phrases identified by our system overlap with their analysis; these are indicated with asterisks. In contrast to our model's success at extracting cue phrases from the meeting dataset, only very common words are selected for the textbook dataset. This may help to explain why cue phrases improve performance for meeting transcripts, but not for the textbook.

7 Conclusions

This paper presents a novel Bayesian approach to unsupervised topic segmentation. Our algorithm is capable of incorporating both lexical cohesion and cue phrase features in a principled manner, and outperforms state-of-the-art baselines on text and transcribed speech corpora. We have developed exact and sampling-based inference techniques, both of which search only over the space of segmentations and marginalize out the associated language models. Finally, we have shown that our model provides a theoretical framework with connections to information theory, while also generalizing and justifying prior work. In the future, we hope to explore the use of similar Bayesian techniques for hierarchical segmentation, and to incorporate additional features such as prosody and speaker change information.

Acknowledgments

The authors acknowledge the support of the National Science Foundation (CAREER grant IIS-0448168) and the Microsoft Research Faculty Fellowship. Thanks to Aaron Adler, S. R. K. Branavan, Harr Chen, Michael Collins, Randall Davis, Dan Roy, David Sontag and the anonymous reviewers for helpful comments and suggestions. We also thank Michel Galley, Igor Malioutov, and Masao Utiyama for making their topic segmentation code publically available. 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

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

José M. Bernardo and Adrian F. M. Smith. 2000. *Bayesian Theory*. Wiley.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

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

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

Micha Elsner and Eugene Charniak. 2008. You Talking to Me? A Corpus and Algorithm for Conversation Disentanglement. In *Proceedings of ACL*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of ACL*, pages 363–370.

Michel Galley, Kathleen R. McKeown, Eric Fosler-Lussier, and Hongyan Jing. 2003. Discourse segmentation of multi-party conversation. *Proceedings of ACL*, pages 562–569.

Jean-Luc Gauvain and Chin-Hui Lee. 1994. Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains. *IEEE Transactions on Speech and Audio Processing*, 2(2):291–298.

Dmitriy Genzel and Eugene Charniak. 2002. Entropy rate constancy in text. In *Proceedings of ACL*, pages 199–206.

Sharon Goldwater and Tom Griffiths. 2007. A fully bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of ACL*, pages 744–751.

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

Barbara Grosz. 1977. The representation and use of focus in dialogue understanding. Technical Report 151, Artificial Intelligence Center, SRI International.

M. A. K. Halliday and Ruqaiya Hasan. 1976. *Cohesion in English*. Longman.

Marti A. Hearst. 1994. Multi-paragraph segmentation of expository text. In *Proceedings of ACL*, pages 9–16.

Julia Hirschberg and Diane Litman. 1993. Empirical studies on the disambiguation of cue phrases. *Computational Linguistics*, 19(3):501–530.

A. Janin, D. Baron, J. Edwards, D. Ellis, D. Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke, et al. 2003. The ICSI Meeting Corpus. *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, 1.

Norman L. Johnson, Samuel Kotz, and N. Balakrishnan. 1997. *Discrete Multivariate Distributions*. Wiley.

Alistair Knott. 1996. *A Data-Driven Methodology for Motivating a Set of Coherence Relations*. Ph.D. thesis, University of Edinburgh.

Diane J. Litman and Rebecca J. Passonneau. 1995. Combining multiple knowledge sources for discourse segmentation. In *Proceedings of the ACL*, pages 108–115.

Igor Malioutov and Regina Barzilay. 2006. Minimum cut model for spoken lecture segmentation. In *Proceedings of ACL*, pages 25–32.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press.

Thomas P. Minka. 2003. Estimating a dirichlet distribution. Technical report, Massachusetts Institute of Technology.

Rebecca Passonneau and Diane Litman. 1993. Intention-based segmentation: Human reliability and correlation with linguistic cues. In *Proceedings of ACL*, pages 148–155.

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

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

M. Purver, T.L. Griffiths, K.P. Körding, and J.B. Tenenbaum. 2006. Unsupervised topic modelling for multi-party spoken discourse. In *Proceedings of ACL*, pages 17–24.

Caroline Sporleder and Mirella Lapata. 2006. Broad coverage paragraph segmentation across languages and domains. *ACM Transactions on Speech and Language Processing*, 3(2):1–35.

Masao Utiyama and Hitoshi Isahara. 2001. A statistical model for domain-independent text segmentation. In *Proceedings of ACL*, pages 491–498.

H. Kenneth Walker, W. Dallas Hall, and J. Willis Hurst, editors. 1990. *Clinical Methods : The History, Physical, and Laboratory Examinations*. Butterworths.

Greg C. G. Wei and Martin A. Tanner. 1990. A monte carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association*, 85(411), September.

A comparison of Bayesian estimators for unsupervised Hidden Markov Model POS taggers

Jianfeng Gao
Microsoft Research
Redmond, WA, USA
jfgao@microsoft.com

Mark Johnson
Brown Univeristy
Providence, RI, USA
Mark_Johnson@Brown.edu

Abstract

There is growing interest in applying Bayesian techniques to NLP problems. There are a number of different estimators for Bayesian models, and it is useful to know what kinds of tasks each does well on. This paper compares a variety of different Bayesian estimators for Hidden Markov Model POS taggers with various numbers of hidden states on data sets of different sizes. Recent papers have given contradictory results when comparing Bayesian estimators to Expectation Maximization (EM) for unsupervised HMM POS tagging, and we show that the difference in reported results is largely due to differences in the size of the training data and the number of states in the HMM. We invesigate a variety of samplers for HMMs, including some that these earlier papers did not study. We find that all of Gibbs samplers do well with small data sets and few states, and that Variational Bayes does well on large data sets and is competitive with the Gibbs samplers. In terms of times of convergence, we find that Variational Bayes was the fastest of all the estimators, especially on large data sets, and that explicit Gibbs sampler (both pointwise and sentence-blocked) were generally faster than their collapsed counterparts on large data sets.

1 Introduction

Probabilistic models now play a central role in computational linguistics. These models define a probability distribution $\mathrm{P}(\boldsymbol{x})$ over structures or analyses $\boldsymbol{x}$. For example, in the part-of-speech (POS) tagging application described in this paper, which involves predicting the part-of-speech tag t_i of each word w_i in the sentence $\boldsymbol{w} = (w_1, \ldots, w_n)$, the structure $\boldsymbol{x} = (\boldsymbol{w}, \boldsymbol{t})$ consists of the words $\boldsymbol{w}$ in a sentence together with their corresponding parts-of-speech $\boldsymbol{t} = (t_1, \ldots, t_n)$.

In general the probabilistic models used in computational linguistics have adjustable parameters $\boldsymbol{\theta}$ which determine the distribution $\mathrm{P}(\boldsymbol{x} \mid \boldsymbol{\theta})$. In this paper we focus on bitag Hidden Markov Models (HMMs). Since our goal here is to compare algorithms rather than achieve the best performance, we keep the models simple by ignoring morphology and capitalization (two very strong cues in English) and treat each word as an atomic entity. This means that the model parameters $\boldsymbol{\theta}$ consist of the HMM state-to-state transition probabilities and the state-to-word emission probabilities.

In virtually all statistical approaches the parameters $\boldsymbol{\theta}$ are chosen or *estimated* on the basis of training data $\boldsymbol{d}$. This paper studies unsupervised estimation, so $\boldsymbol{d} = \boldsymbol{w} = (w_1, \ldots, w_n)$ consists of a sequence of words w_i containing all of the words of training corpus appended into a single string, as explained below.

Maximum Likelihood (ML) is the most common estimation method in computational linguistics. A Maximum Likelihood estimator sets the parameters to the value $\hat{\boldsymbol{\theta}}$ that makes the likelihood $L_{\boldsymbol{d}}$ of the data $\boldsymbol{d}$ as large as possible:

$$\begin{aligned} L_{\boldsymbol{d}}(\boldsymbol{\theta}) &= \mathrm{P}(\boldsymbol{d} \mid \boldsymbol{\theta}) \\ \hat{\boldsymbol{\theta}} &= \arg\max_{\theta} L_{\boldsymbol{d}}(\boldsymbol{\theta}) \end{aligned}$$

In this paper we use the Inside-Outside algorithm, which is a specialized form of Expectation-

Maximization, to find HMM parameters which (at least locally) maximize the likelihood function $L_{\boldsymbol{d}}$.

Recently there is increasing interest in Bayesian methods in computational linguistics, and the primary goal of this paper is to compare the performance of various Bayesian estimators with each other and with EM.

A Bayesian approach uses Bayes theorem to factorize the *posterior distribution* $\mathrm{P}(\boldsymbol{\theta} \mid \boldsymbol{d})$ into the *likelihood* $\mathrm{P}(\boldsymbol{d} \mid \boldsymbol{\theta})$ and the *prior* $\mathrm{P}(\boldsymbol{\theta})$.

$$\mathrm{P}(\boldsymbol{\theta} \mid \boldsymbol{d}) \quad \propto \quad \mathrm{P}(\boldsymbol{d} \mid \boldsymbol{\theta})\, \mathrm{P}(\boldsymbol{\theta})$$

Priors can be useful because they can express preferences for certain types of models. To take an example from our POS-tagging application, most words belong to relatively few parts-of-speech (e.g., most words belong to a single POS, and while there are some words which are both nouns and verbs, very few are prepositions and adjectives as well). One might express this using a prior which prefers HMMs in which the state-to-word emissions are *sparse*, i.e., each state emits few words. An appropriate Dirichlet prior can express this preference.

While it is possible to use Bayesian inference to find a single model, such as the Maximum A Posteriori or MAP value of $\boldsymbol{\theta}$ which maximizes the posterior $\mathrm{P}(\boldsymbol{\theta} \mid \boldsymbol{d})$, this is not necessarily the best approach (Bishop, 2006; MacKay, 2003). Instead, rather than commiting to a single value for the parameters $\boldsymbol{\theta}$ many Bayesians often prefer to work with the full posterior distribution $\mathrm{P}(\boldsymbol{\theta} \mid \boldsymbol{d})$, as this naturally reflects the uncertainty in $\boldsymbol{\theta}$'s value.

In all but the simplest models there is no known closed form for the posterior distribution. However, the Bayesian literature describes a number of methods for approximating the posterior $\mathrm{P}(\boldsymbol{\theta} \mid \boldsymbol{d})$. Monte Carlo sampling methods and Variational Bayes are two kinds of approximate inference methods that have been applied to Bayesian inference of unsupervised HMM POS taggers (Goldwater and Griffiths, 2007; Johnson, 2007). These methods can also be used to approximate other distributions that are important to us, such as the conditional distribution $\mathrm{P}(\boldsymbol{t} \mid \boldsymbol{w})$ of POS tags (i.e., HMM hidden states) $\boldsymbol{t}$ given words $\boldsymbol{w}$.

This recent literature reports contradictory results about these Bayesian inference methods. Johnson (2007) compared two Bayesian inference algorithms, Variational Bayes and what we call here a point-wise collapsed Gibbs sampler, and found that Variational Bayes produced the best solution, and that the Gibbs sampler was extremely slow to converge and produced a worse solution than EM. On the other hand, Goldwater and Griffiths (2007) reported that the same kind of Gibbs sampler produced much better results than EM on their unsupervised POS tagging task. One of the primary motivations for this paper was to understand and resolve the difference in these results. We replicate the results of both papers and show that the difference in their results stems from differences in the sizes of the training data and numbers of states in their models.

It turns out that the Gibbs sampler used in these earlier papers is not the only kind of sampler for HMMs. This paper compares the performance of four different kinds of Gibbs samplers, Variational Bayes and Expectation Maximization on unsupervised POS tagging problems of various sizes. Our goal here is to try to learn how the performance of these different estimators varies as we change the number of hidden states in the HMMs and the size of the training data.

In theory, the Gibbs samplers produce streams of samples that eventually converge on the true posterior distribution, while the Variational Bayes (VB) estimator only produces an approximation to the posterior. However, as the size of the training data distribution increases the likelihood function and therefore the posterior distribution becomes increasingly peaked, so one would expect this variational approximation to become increasingly accurate. Further the Gibbs samplers used in this paper should exhibit reduced mobility as the size of training data increases, so as the size of the training data increases eventually the Variational Bayes estimator should prove to be superior.

However the two point-wise Gibbs samplers investigated here, which resample the label of each word conditioned on the labels of its neighbours (amongst other things) only require $O(m)$ steps per sample (where m is the number of HMM states), while EM, VB and the sentence-blocked Gibbs samplers require $O(m^2)$ steps per sample. Thus for HMMs with many states it is possible to perform one or two orders of magnitude more iterations of the point-wise Gibbs samplers in the same run-time as the other samplers, so it is plausible that they would yield better results.

2 Inference for HMMs

There are a number of excellent textbook presentations of Hidden Markov Models (Jelinek, 1997; Manning and Schütze, 1999), so we do not present them in detail here. Conceptually, a Hidden Markov Model uses a Markov model to generate the sequence of states $\boldsymbol{t} = (t_1, \ldots, t_n)$ (which will be interpreted as POS tags), and then generates each word w_i conditioned on the corresponding state t_i.

We insert endmarkers at the beginning and end of the corpus and between sentence boundaries, and constrain the estimators to associate endmarkers with a special HMM state that never appears elsewhere in the corpus (we ignore these endmarkers during evaluation). This means that we can formally treat the training corpus as one long string, yet each sentence can be processed independently by a first-order HMM.

In more detail, the HMM is specified by a pair of multinomials $\boldsymbol{\theta}_t$ and $\boldsymbol{\phi}_t$ associated with each state t, where $\boldsymbol{\theta}_t$ specifies the distribution over states t' following t and $\boldsymbol{\phi}_t$ specifies the distribution over words w given state t.

$$\begin{array}{rcccl} t_i & \mid & t_{i-1} = t & \sim & \mathrm{Multi}(\boldsymbol{\theta}_t) \\ w_i & \mid & t_i = t & \sim & \mathrm{Multi}(\boldsymbol{\phi}_t) \end{array} \quad (1)$$

The Bayesian model we consider here puts a fixed uniform Dirichlet prior on these multinomials. Because Dirichlets are conjugate to multinomials, this greatly simplifies inference.

$$\begin{array}{rccl} \boldsymbol{\theta}_t & \mid \ \alpha & \sim & \mathrm{Dir}(\boldsymbol{\alpha}) \\ \boldsymbol{\phi}_t & \mid \ \alpha' & \sim & \mathrm{Dir}(\boldsymbol{\alpha}') \end{array}$$

A multinomial $\boldsymbol{\theta}$ is distributed according to the Dirichlet distribution $\mathrm{Dir}(\boldsymbol{\alpha})$ iff:

$$\mathrm{P}(\boldsymbol{\theta} \mid \boldsymbol{\alpha}) \ \propto \ \prod_{j=1}^{m} \theta_j^{\alpha_j - 1}$$

In our experiments we set $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}'$ to the uniform values (i.e., all components have the same value α or α'), but it is possible to estimate these as well (Goldwater and Griffiths, 2007). Informally, α controls the sparsity of the state-to-state transition probabilities while α' controls the sparsity of the state-to-word emission probabilities. As α' approaches zero the prior strongly prefers models in which each state emits as few words as possible, capturing the intuition that most word types only belong to one POS mentioned earlier.

2.1 Expectation Maximization

Expectation-Maximization is a procedure that iteratively re-estimates the model parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$, converging on a local maximum of the likelihood. Specifically, if the parameter estimate at iteration ℓ is $(\boldsymbol{\theta}^{(\ell)}, \boldsymbol{\phi}^{(\ell)})$, then the re-estimated parameters at iteration $\ell + 1$ are:

$$\begin{array}{rcl} \theta^{(\ell+1)}_{t'|t} & = & \mathrm{E}[n_{t',t}]/\mathrm{E}[n_t] \\ \phi^{(\ell+1)}_{w|t} & = & \mathrm{E}[n'_{w,t}]/\mathrm{E}[n_t] \end{array} \quad (2)$$

where $n'_{w,t}$ is the number of times word w occurs with state t, $n_{t',t}$ is the number of times state t' follows t and n_t is the number of occurences of state t; all expectations are taken with respect to the model $(\boldsymbol{\theta}^{(\ell)}, \boldsymbol{\phi}^{(\ell)})$.

The experiments below used the Forward-Backward algorithm (Jelinek, 1997), which is a dynamic programming algorithm for calculating the likelihood and the expectations in (2) in $O(nm^2)$ time, where n is the number of words in the training corpus and m is the number of HMM states.

2.2 Variational Bayes

Variational Bayesian inference attempts to find a function $Q(\boldsymbol{t}, \boldsymbol{\theta}, \boldsymbol{\phi})$ that minimizes an upper bound (3) to the negative log likelihood.

$$\begin{array}{rl} & -\log \mathrm{P}(\boldsymbol{w}) \\ = & -\log \displaystyle\int Q(\boldsymbol{t}, \boldsymbol{\theta}, \boldsymbol{\phi}) \frac{\mathrm{P}(\boldsymbol{w}, \boldsymbol{t}, \boldsymbol{\theta}, \boldsymbol{\phi})}{Q(\boldsymbol{t}, \theta, \phi)} \, d\boldsymbol{t} \, d\boldsymbol{\theta} \, d\boldsymbol{\phi} \\ \leq & -\displaystyle\int Q(\boldsymbol{t}, \boldsymbol{\theta}, \boldsymbol{\phi}) \log \frac{\mathrm{P}(\boldsymbol{w}, \boldsymbol{t}, \boldsymbol{\theta}, \boldsymbol{\phi})}{Q(\boldsymbol{t}, \boldsymbol{\theta}, \boldsymbol{\phi})} \, d\boldsymbol{t} \, d\boldsymbol{\theta} \, d\boldsymbol{\phi} \quad (3) \end{array}$$

The upper bound (3) is called the *Variational Free Energy*. We make a "mean-field" assumption that the posterior can be well approximated by a factorized model Q in which the state sequence $\boldsymbol{t}$ does not covary with the model parameters $\boldsymbol{\theta}, \boldsymbol{\phi}$:

$$\mathrm{P}(\boldsymbol{t}, \boldsymbol{\theta}, \boldsymbol{\phi} \mid \boldsymbol{w}) \ \approx \ Q(\boldsymbol{t}, \boldsymbol{\theta}, \boldsymbol{\phi}) \ = \ Q_1(\boldsymbol{t}) Q_2(\boldsymbol{\theta}, \boldsymbol{\phi})$$

$$\mathrm{P}(t_i|\boldsymbol{w}, \boldsymbol{t}_{-i}, \alpha, \alpha') \quad \propto \quad \left(\frac{n'_{w_i,t_i} + \alpha'}{n_{t_i} + m'\alpha'}\right) \left(\frac{n_{t_i,t_{i-1}} + \alpha}{n_{t_{i-1}} + m\alpha}\right) \left(\frac{n_{t_{i+1},t_i} + \mathrm{I}(t_{i-1} = t_i = t_{i+1}) + \alpha}{n_{t_i} + \mathrm{I}(t_{i-1} = t_i) + m\alpha}\right)$$

Figure 1: The conditional distribution for state t_i used in the pointwise collapsed Gibbs sampler, which conditions on all states $\boldsymbol{t}_{-i}$ *except* t_i (i.e., the counts n do not include t_i). Here m' is the size of the vocabulary, m is the number of HMM states and $\mathrm{I}(\cdot)$ is the indicator function (i.e., equal to one if its argument is true and zero otherwise),

The calculus of variations is used to minimize the KL divergence between the desired posterior distribution and the factorized approximation. It turns out that if the likelihood and conjugate prior belong to exponential families then the optimal Q_1 and Q_2 do too, and there is an EM-like iterative procedure that finds locally-optimal model parameters (Bishop, 2006).

This procedure is especially attractive for HMM inference, since it involves only a minor modification to the M-step of the Forward-Backward algorithm. MacKay (1997) and Beal (2003) describe Variational Bayesian (VB) inference for HMMs. In general, the E-step for VB inference for HMMs is the same as in EM, while the M-step is as follows:

$$\begin{aligned} \tilde{\theta}^{(\ell+1)}_{t'|t} &= f(\mathrm{E}[n_{t',t}] + \alpha)/f(\mathrm{E}[n_t] + m\alpha) \qquad (4) \\ \tilde{\phi}^{(\ell+1)}_{w|t} &= f(\mathrm{E}[n'_{w,t}] + \alpha')/f(\mathrm{E}[n_t] + m'\alpha') \\ f(v) &= \exp(\Psi(v)) \end{aligned}$$

where m' and m are the number of word types and states respectively, Ψ is the digamma function and the remaining quantities are as in (2). This means that a single iteration can be performed in $O(nm^2)$ time, just as for the EM algorithm.

2.3 MCMC sampling algorithms

The goal of Markov Chain Monte Carlo (MCMC) algorithms is to produce a stream of samples from the posterior distribution $\mathrm{P}(\boldsymbol{t} \mid \boldsymbol{w}, \boldsymbol{\alpha})$. Besag (2004) provides a tutorial on MCMC techniques for HMM inference.

A Gibbs sampler is a simple kind of MCMC algorithm that is well-suited to sampling high-dimensional spaces. A Gibbs sampler for $\mathrm{P}(\boldsymbol{z})$ where $\boldsymbol{z} = (z_1, \ldots, z_n)$ proceeds by sampling and updating each z_i in turn from $\mathrm{P}(z_i \mid \boldsymbol{z}_{-i})$, where $\boldsymbol{z}_{-i} = (z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n)$, i.e., all of the z *except* z_i (Geman and Geman, 1984; Robert and Casella, 2004).

We evaluate four different Gibbs samplers in this paper, which vary along two dimensions. First, the sampler can either be *pointwise* or *blocked*. A pointwise sampler resamples a single state t_i (labeling a single word w_i) at each step, while a blocked sampler resamples the labels for all of the words in a sentence at a single step using a dynamic programming algorithm based on the Forward-Backward algorithm. (In principle it is possible to use block sizes other than the sentence, but we did not explore this here). A pointwise sampler requires $O(nm)$ time per iteration, while a blocked sampler requires $O(nm^2)$ time per iteration, where m is the number of HMM states and n is the length of the training corpus.

Second, the sampler can either be *explicit* or *collapsed*. An explicit sampler represents and samples the HMM parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ in addition to the states $\boldsymbol{t}$, while in a collapsed sampler the HMM parameters are integrated out, and only the states $\boldsymbol{t}$ are sampled. The difference between explicit and collapsed samplers corresponds exactly to the difference between the two PCFG sampling algorithms presented in Johnson et al. (2007).

An iteration of the pointwise explicit Gibbs sampler consists of resampling $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ given the state-to-state transition counts $\boldsymbol{n}$ and state-to-word emission counts $\boldsymbol{n}'$ using (5), and then resampling each state t_i given the corresponding word w_i and the neighboring states t_{i-1} and t_{i+1} using (6).

$$\begin{aligned} \boldsymbol{\theta}_t \mid \boldsymbol{n}_t, \boldsymbol{\alpha} &\sim \mathrm{Dir}(\boldsymbol{n}_t + \boldsymbol{\alpha}) \\ \boldsymbol{\phi}_t \mid \boldsymbol{n}'_t, \boldsymbol{\alpha}' &\sim \mathrm{Dir}(\boldsymbol{n}'_t + \boldsymbol{\alpha}') \end{aligned} \qquad (5)$$

$$\mathrm{P}(t_i \mid w_i, \boldsymbol{t}_{-i}, \boldsymbol{\theta}, \boldsymbol{\phi}) \propto \theta_{t_i|t_{i-1}} \phi_{w_i|t_i} \theta_{t_{i+1}|t_i} \qquad (6)$$

The Dirichlet distributions in (5) are non-uniform; $\boldsymbol{n}_t$ is the vector of state-to-state transition counts in $\boldsymbol{t}$ leaving state t in the current state vector $\boldsymbol{t}$, while

$\boldsymbol{n}'_t$ is the vector of state-to-word emission counts for state t. See Johnson et al. (2007) for a more detailed explanation, as well as an algorithm for sampling from the Dirichlet distributions in (5).

The samplers that Goldwater and Griffiths (2007) and Johnson (2007) describe are pointwise collapsed Gibbs samplers. Figure 1 gives the sampling distribution for this sampler. As Johnson et al. (2007) explains, samples of the HMM parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ can be obtained using (5) if required.

The blocked Gibbs samplers differ from the pointwise Gibbs samplers in that they resample the POS tags for an entire sentence at a time. Besag (2004) describes the well-known dynamic programming algorithm (based on the Forward-Backward algorithm) for sampling a state sequence $\boldsymbol{t}$ given the words $\boldsymbol{w}$ and the transition and emission probabilities $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$.

At each iteration the explicit blocked Gibbs sampler resamples $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ using (5), just as the explicit pointwise sampler does. Then it uses the new HMM parameters to resample the states $\boldsymbol{t}$ for the training corpus using the algorithm just mentioned. This can be done in parallel for each sentence in the training corpus.

The collapsed blocked Gibbs sampler is a straight-forward application of the Metropolis-within-Gibbs approach proposed by Johnson et al. (2007) for PCFGs, so we only sketch it here. We iterate through the sentences of the training data, resampling the states for each sentence conditioned on the state-to-state transition counts $\boldsymbol{n}$ and state-to-word emission counts $\boldsymbol{n}'$ for the other sentences in the corpus. This is done by first computing the parameters $\boldsymbol{\theta}^\star$ and $\boldsymbol{\phi}^\star$ of a *proposal HMM* using (7).

$$\begin{aligned} \theta^\star_{t'|t} &= \frac{n_{t',t}+\alpha}{n_t+m\alpha} \\ \phi^\star_{w|t} &= \frac{n'_{w,t}+\alpha'}{n_t+m'\alpha} \end{aligned} \qquad (7)$$

Then we use the dynamic programming sampler described above to produce a *proposal state sequence* $\boldsymbol{t}^\star$ for the words in the sentence. Finally, we use a Metropolis-Hastings accept-reject step to decide whether to update the current state sequence for the sentence with the proposal $\boldsymbol{t}^\star$, or whether to keep the current state sequence. In practice, with all but the very smallest training corpora the acceptance rate is very high; the acceptance rate for all of our collapsed blocked Gibbs samplers was over 99%.

3 Evaluation

The previous section described six different unsupervised estimators for HMMs. In this section we compare their performance for English part-of-speech tagging. One of the difficulties in evaluating unsupervised taggers such as these is mapping the system's states to the gold-standard parts-of-speech. Goldwater and Griffiths (2007) proposed an information-theoretic measure known as the *Variation of Information* (VI) described by Meilă (2003) as an evaluation of an unsupervised tagging. However as Goldwater (p.c.) points out, this may not be an ideal evaluation measure; e.g., a tagger which assigns all words the same single part-of-speech tag does disturbingly well under Variation of Information, suggesting that a poor tagger may score well under VI.

In order to avoid this problem we focus here on evaluation measures that construct an explicit mapping between the gold-standard part-of-speech tags and the HMM's states. Perhaps the most straight-forward approach is to map each HMM state to the part-of-speech tag it co-occurs with most frequently, and use this mapping to map each HMM state sequence $\boldsymbol{t}$ to a sequence of part-of-speech tags. But as Clark (2003) observes, this approach has several defects. If a system is permitted to posit an unbounded number of states (which is not the case here) it can achieve a perfect score on by assigning each word token its own unique state.

We can partially address this by cross-validation. We divide the corpus into two equal parts, and from the first part we extract a mapping from HMM states to the parts-of-speech they co-occur with most frequently, and use that mapping to map the states of the second part of the corpus to parts-of-speech. We call the accuracy of the resulting tagging the *cross-validation accuracy*.

Finally, following Haghighi and Klein (2006) and Johnson (2007) we can instead insist that at most one HMM state can be mapped to any part-of-speech tag. Following these authors, we used a greedy algorithm to associate states with POS tags; the accuracy of the resulting tagging is called the *greedy 1-to-1*

	All − 50	All − 17	120K − 50	120K − 17	24K − 50	24K − 17
EM	0.40527	0.43101	0.29303	0.35202	0.18618	0.28165
VB	0.46123	**0.51379**	0.34679	0.36010	0.23823	0.36599
$\mathrm{GS}_{\mathrm{e,p}}$	0.47826	0.43424	0.36984	0.44125	0.29953	0.36811
$\mathrm{GS}_{\mathrm{e,b}}$	0.49371	0.46568	0.38888	**0.44341**	0.34404	0.37032
$\mathrm{GS}_{\mathrm{c,p}}$	**0.49910***	0.45028	**0.42785**	0.43652	**0.39182**	**0.39164**
$\mathrm{GS}_{\mathrm{c,b}}$	0.49486*	0.46193	0.41162	0.42278	0.38497	0.36793

Figure 2: Average greedy 1-to-1 accuracy of state sequences produced by HMMs estimated by the various estimators. The column heading indicates the size of the corpus and the number of HMM states. In the Gibbs sampler (GS) results the subscript "e" indicates that the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ were explicitly sampled while the subscript "c" indicates that they were integrated out, and the subscript "p" indicates pointwise sampling, while "b" indicates sentence-blocked sampling. Entries tagged with a star indicate that the estimator had not converged after weeks of run-time, but was still slowly improving.

	All − 50	All − 17	120K − 50	120K − 17	24K − 50	24K − 17
EM	0.62115	0.64651	0.44135	0.56215	0.28576	0.46669
VB	0.60484	0.63652	0.48427	0.36458	0.35946	0.36926
$\mathrm{GS}_{\mathrm{e,p}}$	0.64190	0.63057	0.53571	0.46986	0.41620	0.37165
$\mathrm{GS}_{\mathrm{e,b}}$	**0.65953**	0.65606	0.57918	0.48975	0.47228	0.37311
$\mathrm{GS}_{\mathrm{c,p}}$	0.61391*	**0.67414**	**0.65285**	**0.65012**	**0.58153**	**0.62254**
$\mathrm{GS}_{\mathrm{c,b}}$	0.60551*	0.65516	0.62167	0.58271	0.55006	0.58728

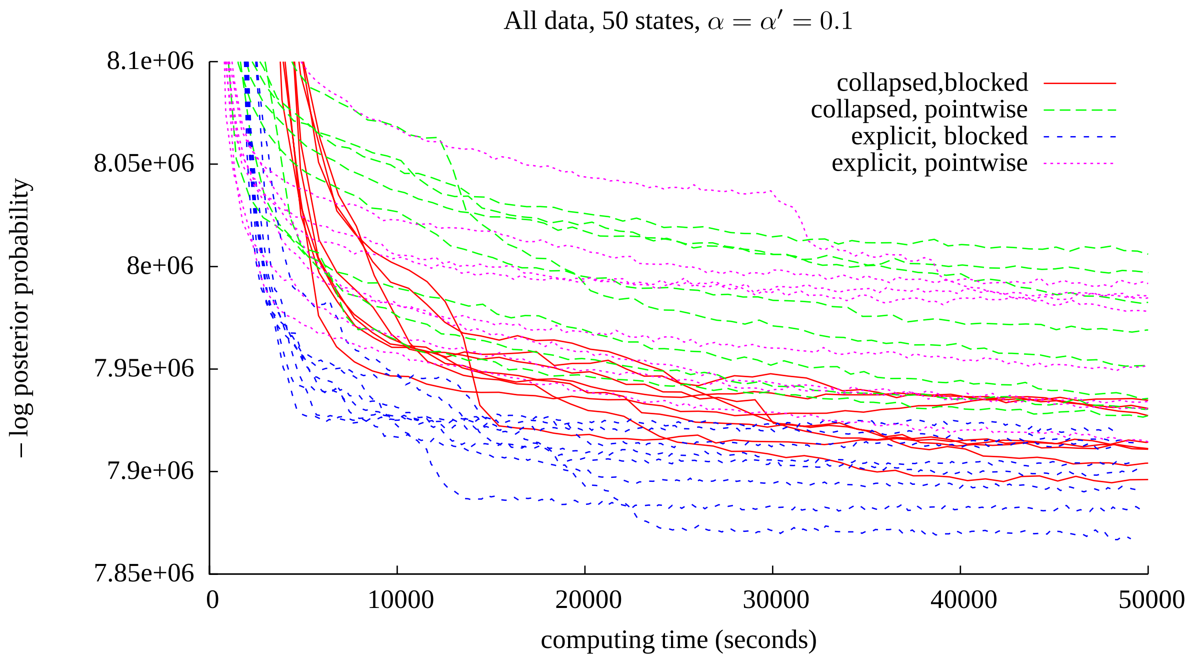
Figure 3: Average cross-validation accuracy of state sequences produced by HMMs estimated by the various estimators. The table headings follow those used in Figure 2.

	All − 50	All − 17	120K − 50	120K − 17	24K − 50	24K − 17
EM	4.47555	3.86326	6.16499	4.55681	7.72465	5.42815
VB	4.27911	3.44029	5.00509	3.19670	4.80778	3.14557
$\mathrm{GS}_{\mathrm{e,p}}$	4.24919	3.53024	4.30457	3.23082	**4.24368**	3.17076
$\mathrm{GS}_{\mathrm{e,b}}$	4.04123	**3.46179**	4.22590	3.20276	4.29474	**3.10609**
$\mathrm{GS}_{\mathrm{c,p}}$	**4.03886***	3.52185	**4.21259**	**3.17586**	4.30928	3.18273
$\mathrm{GS}_{\mathrm{c,b}}$	4.11272*	3.61516	4.36595	3.23630	4.32096	3.17780

Figure 4: Average Variation of Information between the state sequences produced by HMMs estimated by the various estimators and the gold tags (smaller is better). The table headings follow those used in Figure 2.

	All − 50	All − 17	120K − 50	120K − 17	24K − 50	24K − 17
EM	558	346	648	351	**142**	125
VB	**473**	**123**	**337**	**24**	183	**20**
$\mathrm{GS}_{\mathrm{e,p}}$	2863	382	3709	63	2500	177
$\mathrm{GS}_{\mathrm{e,b}}$	3846	286	5169	154	4856	139
$\mathrm{GS}_{\mathrm{c,p}}$	$\star$	34325	44864	40088	45285	43208
$\mathrm{GS}_{\mathrm{c,b}}$	$\star$	6948	7502	7782	7342	7985

Figure 5: Average number of iterations until the negative logarithm of the posterior probability (or likelihood) changes by less than 0.5% (smaller is better) per at least 2,000 iterations. No annealing was used.



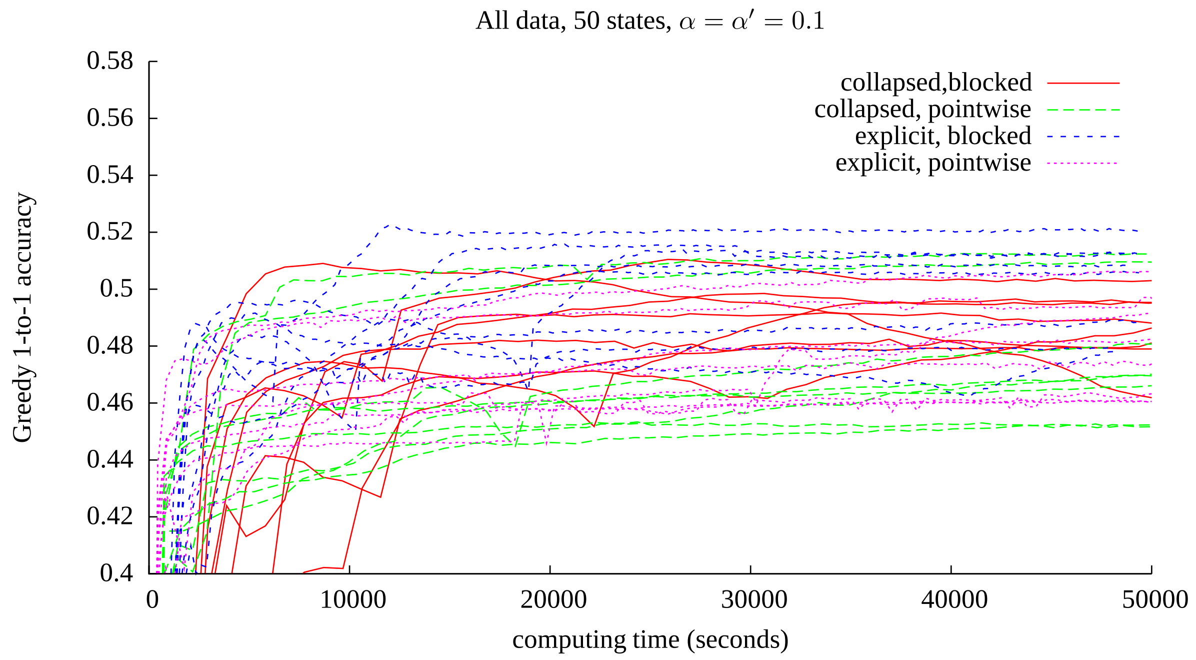


Figure 6: Variation in (a) negative log likelihood and (b) 1-to-1 accuracy as a function of running time on a 3GHz dual quad-core Pentium for the four different Gibbs samplers on all data and 50 hidden states. Each iteration took approximately 96 sec. for the collapsed blocked sampler, 7.5 sec. for the collapsed pointwise sampler, 25 sec. for the explicit blocked sampler and 4.4 sec. for the explicit pointwise sampler.

accuracy.

The studies presented by Goldwater and Griffiths (2007) and Johnson (2007) differed in the number of states that they used. Goldwater and Griffiths (2007) evaluated against the reduced tag set of 17 tags developed by Smith and Eisner (2005), while Johnson (2007) evaluated against the full Penn Treebank tag set. We ran all our estimators in both conditions here (thanks to Noah Smith for supplying us with his tag set).

Also, the studies differed in the size of the corpora used. The largest corpus that Goldwater and Griffiths (2007) studied contained 96,000 words, while Johnson (2007) used all of the 1,173,766 words in the full Penn WSJ treebank. For that reason we ran all our estimators on corpora containing 24,000 words and 120,000 words as well as the full treebank.

We ran each estimator with the eight different combinations of values for the hyperparameters α and α' listed below, which include the optimal values for the hyperparameters found by Johnson (2007), and report results for the best combination for each estimator below [1].

α	α'
1	1
1	0.5
0.5	1
0.5	0.5
0.1	0.1
0.1	0.0001
0.0001	0.1
0.0001	0.0001

Further, we ran each setting of each estimator at least 10 times (from randomly jittered initial starting points) for at least 1,000 iterations, as Johnson (2007) showed that some estimators require many iterations to converge. The results of our experiments are summarized in Figures 2–5.

[1]We found that on some data sets the results are sensitive to the values of the hyperparameters. So, there is a bit uncertainty in our comparison results because it is possible that the values we tried were good for one estimator and bad for others. Unfortunately, we do not know any efficient way of searching the optimal hyperparameters in a much wider and more fine-grained space. We leave it to future work.

4 Conclusion and future work

As might be expected, our evaluation measures disagree somewhat, but the following broad tendancies seem clear. On small data sets all of the Bayesian estimators strongly outperform EM (and, to a lesser extent, VB) with respect to all of our evaluation measures, confirming the results reported in Goldwater and Griffiths (2007). This is perhaps not too surprising, as the Bayesian prior plays a comparatively stronger role with a smaller training corpus (which makes the likelihood term smaller) and the approximation used by Variational Bayes is likely to be less accurate on smaller data sets.

But on larger data sets, which Goldwater et al did not study, the results are much less clear, and depend on which evaluation measure is used. Expectation Maximization does surprisingly well on larger data sets and is competitive with the Bayesian estimators at least in terms of cross-validation accuracy, confirming the results reported by Johnson (2007).

Variational Bayes converges faster than all of the other estimators we examined here. We found that the speed of convergence of our samplers depends to a large degree upon the values of the hyperparameters α and α', with larger values leading to much faster convergence. This is not surprising, as the α and α' specify how likely the samplers are to consider novel tags, and therefore directly influence the sampler's mobility. However, in our experiments the best results are obtained in most settings with small values for α and α', usually between 0.1 and 0.0001.

In terms of time to convergence, on larger data sets we found that the blocked samplers were generally faster than the pointwise samplers, and that the explicit samplers (which represented and sampled $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$) were faster than the collapsed samplers, largely because the time saved in not computing probabilities on the fly overwhelmed the time spent resampling the parameters.

Of course these experiments only scratch the surface of what is possible. Figure 6 shows that pointwise-samplers initially converge faster, but are overtaken later by the blocked samplers. Inspired by this, one can devise hybrid strategies that interleave blocked and pointwise sampling; these might perform better than both the blocked and pointwise samplers described here.

References

Matthew J. Beal. 2003. *Variational Algorithms for Approximate Bayesian Inference*. Ph.D. thesis, Gatsby Computational Neuroscience unit, University College London.

Julian Besag. 2004. An introduction to Markov Chain Monte Carlo methods. In Mark Johnson, Sanjeev P. Khudanpur, Mari Ostendorf, and Roni Rosenfeld, editors, *Mathematical Foundations of Speech and Language Processing*, pages 247–270. Springer, New York.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

Alexander Clark. 2003. Combining distributional and morphological information for part of speech induction. In *10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 59–66. 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.

Sharon Goldwater and Tom Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 744–751, Prague, Czech Republic, June. Association for Computational Linguistics.

Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 320–327, New York City, USA, June. Association for Computational Linguistics.

Frederick Jelinek. 1997. *Statistical Methods for Speech Recognition*. The MIT Press, Cambridge, Massachusetts.

Mark Johnson, Thomas Griffiths, and Sharon Goldwater. 2007. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 139–146, Rochester, New York, April. Association for Computational Linguistics.

Mark Johnson. 2007. Why doesn't EM find good HMM POS-taggers? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 296–305.

David J.C. MacKay. 1997. Ensemble learning for hidden Markov models. Technical report, Cavendish Laboratory, Cambridge.

David J.C. MacKay. 2003. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press.

Chris Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts.

Marina Meilă. 2003. Comparing clusterings by the variation of information. In Bernhard Schölkopf and Manfred K. Warmuth, editors, *COLT 2003: The Sixteenth Annual Conference on Learning Theory*, volume 2777 of *Lecture Notes in Computer Science*, pages 173–187. Springer.

Christian P. Robert and George Casella. 2004. *Monte Carlo Statistical Methods*. Springer.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 354–362, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Transliteration as Constrained Optimization

Dan Goldwasser **Dan Roth**
Department of Computer Science
University of Illinois
Urbana, IL 61801
{goldwas1,danr}@uiuc.edu

Abstract

This paper introduces a new method for identifying named-entity (NE) transliterations in bilingual corpora. Recent works have shown the advantage of discriminative approaches to transliteration: given two strings (w_s, w_t) in the source and target language, a classifier is trained to determine if w_t is the transliteration of w_s. This paper shows that the transliteration problem can be formulated as a constrained optimization problem and thus take into account contextual dependencies and constraints among character bi-grams in the two strings. We further explore several methods for learning the objective function of the optimization problem and show the advantage of learning it discriminately. Our experiments show that the new framework results in over 50% improvement in translating English NEs to Hebrew.

1 Introduction

Named entity (NE) transliteration is the process of transcribing a NE from a source language to some target language based on phonetic similarity between the entities. Identifying transliteration pairs is an important component in many linguistic applications which require identifying out-of-vocabulary words, such as machine translation and multilingual information retrieval (Klementiev and Roth, 2006b; Hermjakob et al., 2008).

It may appear at first glance that identifying the phonetic correlation between names based on an orthographic analysis is a simple, straight-forward task; however in many cases a consistent deterministic mapping between characters does not exist; rather, the mapping depends on the context the characters appear in and on transliteration conventions which may change across domains. Figure 1 exhibits two examples of NE transliterations in English and Hebrew, with the correct mapping across the two scripts. Although the two Hebrew names share a common prefix[1], this prefix can be mapped into a single English character or into two different characters depending on the context it appears in. Similarly, depending on the context it appears in, the English character a can be mapped into different characters or to an "empty" character.

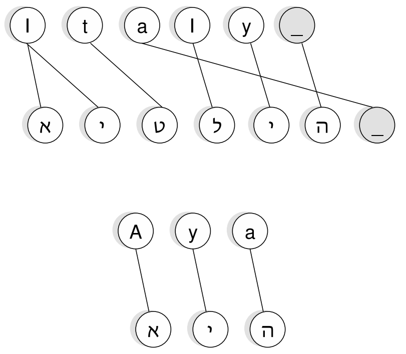


Figure 1: Named entities transliteration pairs in English and Hebrew and the character level mapping between the two names. The Hebrew names can be romanized as *ee-ta-l-ya* and *a-ya*

[1]In all our example the Hebrew script is shown left-to-right to simplify the visualization of the transliteration mapping.

In recent years, as it became clear that solutions that are based on linguistics rules are not satisfactory, machine learning approaches have been developed to address this problem. The common approach adopted is therefore to view this problem as a classification problem (Klementiev and Roth, 2006a; Tao et al., 2006) and train a discriminative classifier. That is, given two strings, one in the source and the other in the target language, extract pairwise features, and train a classifier that determines if one is a transliteration of the other. Several papers have followed up on this basic approach and focused on semi-supervised approaches to this problem or on extracting better features for the discriminative classifier (Klementiev and Roth, 2006b; Bergsma and Kondrak, 2007; Goldwasser and Roth, 2008). While it has been clear that the relevancy of pairwise features is context sensitive and that there are contextual constraints among them, the hope was that a discriminative approach will be sufficient to account for those by weighing features appropriately. This has been shown to be difficult for language pairs which are very different, such as English and Hebrew (Goldwasser and Roth, 2008).

In this paper, we address these difficulties by proposing to view the transliteration decision as a globally phrased constrained optimization problem. We formalize it as an optimization problem over a set of local pairwise features – character n-gram matches across the two string – and subject to legitimacy constraints.

We use a discriminatively trained classifier as a way to learn the objective function for the global constrained optimization problem. Our technical approach follows a large body of work developed over the last few years, following (Roth and Yih, 2004) that has formalized global decisions problems in NLP as constrained optimization problems and solved these optimization problems using Integer Linear Programming (ILP) or other methods (Punyakanok et al., 2005; Barzilay and Lapata, 2006; Clarke and Lapata, ; Marciniak and Strube, 2005).

We investigate several ways to train our objective function, which is represented as a dot product between a set of features chosen to represent a pair (w_s, w_t), and a vector of initial weights. Our first baseline makes use of all features extracted from a pair, along with a simple counting method to determine initial weights. We then use a method similar to (Klementiev and Roth, 2006a; Goldwasser and Roth, 2008) in order to discriminatively train a better weight vector for the objective function.

Our key contribution is that we use a constrained optimization approach also to determine a better feature representation for a given pair. (Bergsma and Kondrak, 2007) attempted a related approach to restricting the set of features representing a transliteration candidate. However, rather than directly aligning the two strings as done there, we exploit the expressiveness of the ILP formulation and constraints to generate a better representation of a pair. This is the representation we then use to discriminatively learn a better weight vector for the objective function used in our final model.

Our experiments focus on Hebrew-English transliteration, which were shown to be very difficult in a previous work (Goldwasser and Roth, 2008). We show very significant improvements over existing work with the same data set, proving the advantage of viewing the transliteration decision as a global inference problem. Furthermore, we show the importance of using a discriminatively trained objective function.

The rest of the paper is organized as follows. The main algorithmic contribution of this paper is described in Sec. 2. Our experimental study is describes in Sec. 3 and Sec. 4 concludes.

2 Using inference for transliteration

In this section we present our transliteration decision framework, which is based on solving a constrained optimization problem with an objective function that is discriminatively learned. Our framework consists of three key elements:

1. **Decision Model** When presented with a NE in the source language w_s and a set of candidates $\{w_t\}_1^k$ in the target language, the decision model ranks the candidate pairs (w_s, w_t) and selects the "best" candidate pair. This is framed as an optimization problem

$$w_t^* = argmax_i\{w \cdot F(w_s, w_t^i)\}, \qquad (1)$$

where F is a feature vector *representation* of the pair (w_s, w_t^i) and w is a vector of *weights* assigned to each feature.

2. **Representation** A pair $s = (w_s, w_t)$ of source and target NEs is represented as a vector of *features*, each of which is a pair of character n-grams, from w_s and w_t, resp. Starting with a baseline representation introduced in (Klementiev and Roth, 2006a), denoted here $AF(s)$, we refine this representation to take into account dependencies among the individual n-gram pairs. This refinement process is framed as a constrained optimization problem:

$$F(s)^* = argmax_{F \subseteq AF}\{w \cdot AF(s)\}, \quad (2)$$

subject to a set C of linear constraints. Here AF is the initial representation (All$-$Features), w is a vector of *weights* assigned to each feature and C is a set of constraints accounting for interdependencies among features.

3. **Weight Vector** Each pairwise n-gram feature is associated with a weight; this weigh vector is used in both optimization formulations above. The weight vector is determined by considering the whole training corpus. The initial weight vector is obtained generatively, by counting the relative occurrence of substring pairs in positive examples. The representation is refined by discriminatively training a classifier to maximize transliteration performance on the training data. In doing that, each example is represented using the feature vector representation described above.

The three key operations described above are being used in several stages, with different parameters (weight vectors and representations) as described in Alg. 1. In each stage a different element is refined. The input to this process is a training corpus Tr=(D_S,D_T) consisting of NE transliteration pairs $s = (w_s, w_t)$, where w_s, w_t are NEs in the source and target language, respectively. Each such sample point is initially represented as a feature vector $AF(s)$ (for All$-$Features), where features are pairs of substrings from the two words (following (Klementiev and Roth, 2006a)).

Given the set of feature vectors generated by applying AF to Tr, we assign initial weights W to the features ((1) in Alg. 1). These weights form the initial objective function used to construct a new feature based representation, Informative$-$Features, $IF_W(s)$ ((2) in Alg. 1). Specifically, for an instance s, $IF_W(s)$ is the solution of the optimization problem in Eq. 2, with W as the weight vector, $AF(s)$ as the representation, and a set of constraints ensuring the "legitimacy" of the selected set of features (Sec. 2.2.1).

```
Input: Training Corpora Tr=($D_S$,$D_T$)
Output: Transliteration model $\mathcal{M}$

1. Initial Representation and Weights
    For each sample s ∈ Tr, use AF to generate a feature vector
    $\{(f_s, f_t)^1, (f_s, f_t)^2, \ldots, (f_s, f_t)^n\} \in \{0,1\}^n$.
    Define $W{:}f \rightarrow \mathcal{R}$ s.t. foreach feature $f=(f_s, f_t)$
        $W(f) = \frac{\#(f_s,f_t)}{\#(f_s)} \times \frac{\#(f_s,f_t)}{\#(f_t)}$
2. Inferring Informative Representation ($W$)
    Modify the initial representation by solving the following constrained optimization problem:
    $IF_W(s)^* = argmax_{IF(s) \subseteq (AF(s))} W \cdot AF(s)$,
    subject to constraints C.
3. Discriminative Training
    Train a discriminative model on Tr, using $\{IF(s)\}_{s \in Tr}$.
    Let $W_D$ be the new weight vector obtained by discriminative training.
4. Inferring Informative Representation ($W_D$)
    Modify the initial representation by solving the following constrained optimization problem. This time, the objective function is determined by the discriminatively trained weight vector $W_D$.
    $IF_{W_D}(s)^* = argmax_{IF(s) \subseteq (AF(s))} W_D \cdot AF(s)$,
    subject to constraints C.
5. Decision Model
    Given a word $w_s$ and a list of candidates $w_t^1, w_t^2, \ldots w_t^k$, the chosen transliteration is $w_{t^*}$, determined by:
        $t^* = argmax_i\{W_D \cdot IF_{W_D}((w_s, w_t^i))\}$
```

Algorithm 1: Transliteration Framework.

The new feature extraction operator $IF_W(s)$ is now used to construct a new representation of the training corpus. With this representation, we train discriminately a new weight vector W_D. This weight vector, now defines a new objective function for the optimization problem in Eq. 2; W_D is the weight vector and $AF(s)$ the representation. We denote by $IF_{W_D}(s)$ the solution of this optimization problem for an instance s.

Given a representation and a weight vector, the optimization problem in Eq. 1 is used to find the transliteration of w_s. Our best decision model makes use of Eq. 1 using W_D as the feature vector and $IF_{W_D}(s)$ as the feature representation of s.

The rest of this section provides details on the operations and how we use them in different stages.

2.1 Initial Representation and Weights

The feature space we consider consists of n potential features, each feature $f = (f_s, f_t)$ represents a pairing of character level n-grams, where $f_s \in \{$*Source-Language* $\cup$ *empty-string* $\}$ and $f_t \in \{$*Target-Language* $\cup$ *empty-string*$\}$. A given sample (w_s, w_t) consisting of a pair of NEs is represented as a features vector $s \in \{0,1\}^n$. We say that a feature f^i is active if $f^i = 1$ and that $s_1 \subset s_2, \iff \{f^i\}_{\{f^i=\,1\ in\ s_1\}} \subset \{f^i\}_{\{f^i=1\ in\ s_2\}}$. We represent the active features corresponding to a pair as a bipartite graph $G = (V, E)$, in which each vertex $v \in V$ either represents the empty string, a single character or a bi-gram. V^S, V^T denote the vertices representing source and target language n-grams respectively. Each of these sets is composed of two disjoint subsets: $V_S = V^S_U \cup V^S_B$, $V_T = V^T_U \cup V^T_B$ consisting of vertices representing the uni-gram and bi-gram strings. Given a vertex v, *degree(*v, V'*)*denotes the degree of v in a subgraph of G, consisting only of $V' \subset V$; *index(v)* is the index of the substring represented by v in the original string.

Edges in the bipartite graph represent active features. The only deviation is that the vertex representing the empty string can be connected to any other (non-empty) vertex.

Our initial feature extraction method follows the one presented in (Klementiev and Roth, 2006a), in which the feature space consists of n-gram pairs from the two languages. Given a pair, each word is decomposed into a set of character substrings of up to a given length (including the empty string). Features are generated by pairing substrings from the two sets whose relative positions in the original words differ by k or less places, or formally:

$$E = \{e = (v_i, v_j) \mid (v_i \in V_S \wedge v_j \in V_T) \Rightarrow (index(v_j) + k \geq index(v_i) \geq index(v_j) - k) \wedge (v_i \neq v_{empty-string} \vee v_j \neq v_{empty-string})\}.$$

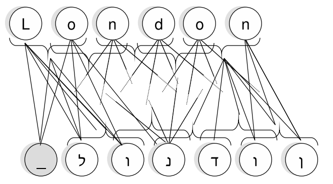


Figure 2: All possible unigram and bigram pairs generated by the AF operator. The Hebrew name can be romanized as *lo-n-do-n*

In our experiments we used *k*=1 which tested empirically, achieved the best performance.

Figure 2 exhibits the active features in the example using the graph representation. We refer to this feature extraction method as $All\text{-}Features$ (AF), and define it formally as an operator $AF : s \rightarrow \{(f_s, f_t)^i\}$ that maps a sample point $s = (w_s, w_t)$ to a set of active features.

The initial sample representation generates features by coupling substrings from the two terms without considering the dependencies between the possible consistent combinations. Ideally, given a positive sample, it is desirable that paired substrings would encode phonetic similarity or a distinctive context in which the two substrings correlate. However, AF simply pairs substrings from the two words, resulting in a noisy representation of the sample point. Given enough positive samples, we assume that features appearing with distinctive frequency will encode the desired relation. We use this observation, and construct a weight vector, associating each feature with a positive number indicating its relative occurrence frequency in the training data representation formed by AF. This weight is computed as follows:

Definition 1 (Initial Feature Weights Vector) *Let* W*:*f $\rightarrow \mathcal{R}$ *s.t. for each feature* f=$\{f_s, f_t\}$,

$$W(\text{f}) = \frac{\#(f_s, f_t)}{\#(f_s)} \times \frac{\#(f_s, f_t)}{\#(f_t)},$$

where $\#(f_s, f_t)$ *is the number of occurrences of that feature in the positive sample set, and* $\#(f_L), L = \{s, t\}$ *is the number of occurrences of an individual substring, in any of the features extracted from positive samples in the training set.*

These weights transform every example into a weighted graph, where each edge is associated by W with the weight assigned to the feature it represents. As we empirically tested, this initialization assigns high weights to features that preserve the phonetic correlation between the two languages. The top part of figure 5 presents several examples of weights assigned by W to features composed of different English and Hebrew substrings combinations. It can be observed that combination which are phonetically similar are associated with a higher weight. However, as it turns out, transliteration mappings do not consist of "clean" and consistent mappings of phonetically similar substrings. In the following section we explain how to use these weights to generate a more compact representation of samples.

2.2 Inferring Informative Representations

In this section we suggest a new feature extraction method for determining the representation of a given word pair. We use the strength of the active features computed above, along with legitimacy constraints on mappings between source and target strings to find an optimal set of consistent active features that represents a pair. This problem can be naturally encoded as a linear optimization problem, which seeks to maximize a linear objective function determined by W, over a set of variables representing the active features selection, subject to a set of linear constraints representing the dependencies between selections. We follow the formulation given by (Roth and Yih, 2004), and define it as an Integer Linear Programming (ILP) optimization problem, in which each integer variable $a_{(j,k)}$, defined over $\{0,1\}$, represents whether a feature pairing an n-gram $j \in S$ with an n-gram $k \in T$, is active. Although using ILP is in general NP-hard, it has been used efficiently in many natural language (see section 1). Our experience as well has been that this process is very efficient due to the sparsity of the constraints used.

2.2.1 Constraining Feature Dependencies

To limit the selection of active features in each sample we require that each element in the decomposition of w_s into bi-grams should be paired with an element in w_t, and the vice-versa. We restrict the possible pairs by allowing only a single n-gram to be matched to any other n-gram, with one exception - we allow every bi-gram to be mapped into an empty string. Viewed as a bipartite graph, we allow each node (with the exception of the empty string) to have only one connected edge. These constraints, given the right objective function, should enforce an alignment of bi-grams according to phonetic similarity; for example, the word pairs described in Figure 1, depicts a character level alignment between the words, where in some cases a bi-gram is mapped into a single character and in other cases single characters are mapped to each other, based on phonetic similarity encoded by the two scripts. However, imposing these constraints over the entire set of candidate features would be too restrictive; it is unlikely that one can consistently represent a single "correct" phonetic mapping. We wish to represent both the character level and bi-gram mapping between names as both represent informative features on the correspondence between the names over the two scripts. To allow this, we decompose the problem into two disjoint sets of constraints imposing 1-1 mappings, one over the set of single character substrings and the other over the bi-gram substrings. Given the bipartite graph generated by *AF*, we impose the following constraints:

Definition 2 (Transliteration Constraints) *Let C be the set of constraints, consisting of the following predicates:*

$$\forall v \in V^S,\ \text{degree}(v, V^S \cup V_U^T) \leq 1 \wedge$$
$$\forall v \in V^S,\ \text{degree}(v, V^S \cup V_B^T) \leq 1 \wedge$$
$$\forall v \in V^T,\ \text{degree}(v, V^T \cup V_U^S) \leq 1 \wedge$$
$$\forall v \in V^T,\ \text{degree}(v, V^T \cup V_B^S) \leq 1$$

For example, Figure 2 shows the graph of all possible candidates produced by AF. In Figure 3, the graph is decomposed into two graphs, each depicting possible matches between the character level uni-gram or bi-gram substrings. the ILP constraints ensure that in each graph, every node (with the exception of the empty string) has a degree of one . Figure 4 gives the results of the ILP process – a unified graph in which every node has only a single edge associated with it.

Definition 3 (Informative Feature Extraction (IF)) *We define the Informative-Features(IF) feature extraction operator, $IF : s \rightarrow \{(f_s, f_t)^i\}$ as the solution to the ILP problem in Eq. 2. Namely,*

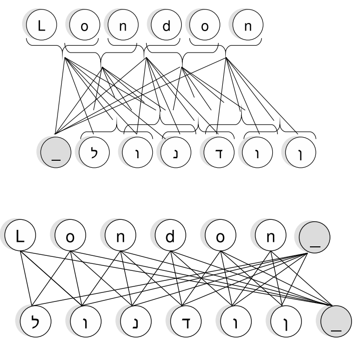


Figure 3: Find informative features by solving an ILP problem. Dependencies between matching decisions are modeled by allowing every node to be connected to a single edge (except the node representing the empty-string).

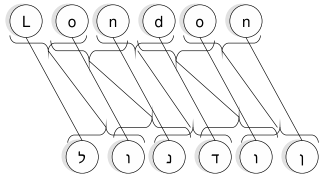


Figure 4: The result of applying the IF operator by solving an ILP problem, represented as a pruned graph.

$$IF(s)^{*} = argmax_{IF(s)\subseteq(AF(s))} w \cdot AF(s),$$

subject to constraints ***C***.

We will use this operator with $w = W$, defined above, and denote it IF_W, and also use it with a different weight vector, trained discriminatively, as described next.

2.3 Discriminative Training

Using the IF_W operator, we generate a better representation of the training data, which is now used to train a discriminative model. We use a linear classifier trained with a regularized average perceptron update rule (Grove and Roth, 2001) as implemented in SNoW, (Roth, 1998). This learning algorithm provides a simple and general linear classifier that has been demonstrated to work well in other NLP classification tasks, e.g. (Punyakanok et al., 2005), and allows us to incorporate extensions such as strength of features naturally into the training algorithm. We augment each sample in the training data with feature weights; given a sample, the learner is presented with a real-valued feature vector instead of a binary vector. This can be viewed as providing a better starting point for the learner, which improves the learning rate (Golding and Roth, 1999; Ng and Jordan, 2001).

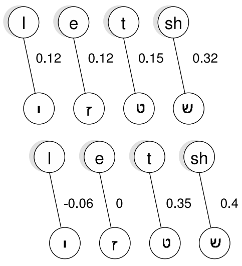


Figure 5: Several examples of weights assigned to features generated by coupling English and Hebrew substrings. Top figure: initial weights. Bottom figure: Discriminatively learned weights. The Hebrew characters, ordered left to right, can be romanized as *y,z,t,sh*

The weight vector learned by the discriminative training is denoted W_D. Given the new weight vector, we can define a new feature extraction operator, that we get by applying the objective function in Eq. 2 with W_D instead of W. Given a sample s, the feature representation generated by this new information extraction operator is denoted $IF_{W_D}(s)$. The key difference between W and W_D is that the latter was trained over a corpora containing both negative and positive examples, and as a result W_D contains negative weights. To increase the impact of training we multiplied the negative weights by 2.

Figure 5 presents some examples of the benefit of discriminately learning the objective function; the weighted edges in the top figure show the values assigned to features by W, while the bottom figure shows the weights assigned by W_D. In all cases, phonetically similar characters were assigned higher scores by W_D, and character pairs not phonetically similar were typically assigned negative weights. It is also interesting to note a special phenomena occurring in English-Hebrew transliterations. The English vowels will be paired to almost any Hebrew character when generating pairs using AF, since vowels in most cases are omitted in Hebrew, there is no distinctive context in which English vowels appear. We can see for example, in the top graph presented in Figure 5 an edge matching a vowel to a Hebrew character with a high weight, the bottom graph showing the results of the discriminative training process show that this edge is associated with a zero weight score.

2.4 Decision Models

This section defines several transliteration decision models given a word w_s and a list of candidates $w_t^1, w_t^2, \dots w_t^k$. The models are used to identify the correct transliteration pair from the set of candidates $\{s_i = (w_s, w_t^i)\}_{i=1\dots k}$.

In all cases, the decision is formulated as in Eq. 1, where different models differ by the representations and weight vectors used.

Decision Model 1 *Ranking the transliteration candidates is done by evaluating*

$$s^* = argmax_i \; W \cdot AF(s_i),$$

which selects the transliteration pair which maximizes the objective function based on the generatively computed weight vector.

Decision Model 2 *Ranking the transliteration candidates is done by evaluating:*

$$s^* = argmax_i \; W_D \cdot AF(s_i)).$$

This decision model is essentially equivalent to the transliteration models used in (Klementiev and Roth, 2006a; Goldwasser and Roth, 2008), in which a linear transliteration model was trained using a feature extraction method equivalent to *AF*.

Decision Model 3 *Ranking the transliteration candidates is done by evaluating:*

$$s^* = argmax_i \; W \cdot IF_W(s_i),$$

which maximizes the objective function with the generatively computed weight vector and the informative feature representation derived based on it.

Decision Model 4 *Ranking the transliteration candidates is done by evaluating:*

$$s^* = argmax_i \; W_D \cdot IF_W(s_i)),$$

which conceptually resembles the transliteration model presented in (Bergsma and Kondrak, 2007), in that a discriminative classifier was trained and used over a pruned feature set.

Decision Model 5 *Ranking the transliteration candidates is done by evaluating:*

$$s^* = argmax_i \; W_D \cdot IF_{W_D}(s_i),$$

which maximize the objective function with the discriminately derived weight vector and the informative features inferred based on it. This decision model is the only model that incorporates discriminative weights as part of the feature extraction process; W_D is used as the objective function used when inferring IF_{W_D}.

3 Evaluation

We evaluated our approach over a corpus of 300 English-Hebrew transliteration pairs, and used another 250 different samples for training the models. We constructed the test set by pairing each English name with all Hebrew names in the corpus. The system was evaluated on its ability to correctly identify the 300 transliteration pairs out of all the possible transliteration candidates. We measured performance using the *Mean Reciprocal Rank* (MRR) measure. This measure, originally introduced in the field of information retrieval, is used to evaluate systems that rank several options according to their probability of correctness. MRR is a natural measure in our settings and has been used previously for evaluating transliteration systems, for example by (Tao et al., 2006).

Given a set Q of queries and their respective responses ranked according to the system's confidence, we denote the rank of the correct response to a query $q_i \in Q$ as $rank(q_i)$. MRR is then defined as the average of the multiplicative inverse of the rank of the correct answer, that is:

$$MRR = \frac{1}{|Q|} \sum_{i=1\dots|Q|} \frac{1}{rank(q_i)}.$$

In our experiments we solved an ILP problem for every transliteration candidate pairs, and computed MRR with respect to the confidence of our decision model across the candidates. Although this required solving thousands of ILP instances, it posed no computational burden as these instances typically contained a small number of variables and constraints. The entire test set is solved in less than 20 minutes using the publicly available GLPK package (`http://www.gnu.org/software/glpk/`).

The performance of the different models is summarized in table 1, these results are based on a training set of 250 samples used to train the discriminative transliteration models and also to construct the initial weight vector W. Figure 6 shows performance over different number of training examples. Our evaluation is concerns with the core transliteration and decision models presented here and does not consider any data set optimizations that were introduced in previous works, which we view as orthogonal additions, hence the difference with the results published in (Goldwasser and Roth, 2008).

The results clearly show that our final model, model 5, outperform other models. Interestingly, model 1, a simplistic model, significantly outperforms the discriminative model presented in (Klementiev and Roth, 2006b). We believe that this is due to two reasons. It shows that discriminative training over the representation obtained using AF is not efficient; moreover, this phenomenon is accentuated given that we train over a very small data set, which favors generative estimation of weights. This is also clear when comparing the performance of model 1 to model 4, which shows that learning over the representation obtained using constrained optimization (IF) results in a very significant performance improvement.

The improvement of using IF_W is not automatic. Model 3, which uses IF_W, and model 1, which uses AF, converge to nearly the same result. Both these models use generative weights to make the transliteration decision, and this highlights the importance of discriminative training. Both model 4 and model 5 use discriminatively trained weights and significantly outperform model 3. These results indicate that using constraint optimization to generate the examples' representation in itself may not help; the objective function used in this inference has a significant role in improved performance.

The benefit of discriminatively training the objective function becomes even clearer when comparing the performance of model 5 to that of model 4, which uses the original weight vector when inferring the sample representation.

It can be assumed that this algorithm can benefit from further iterations – generating a new feature representations, training a model on it, and using the resulting model as a new objective function. However, it turns out that after a single round, improved weights due to additional training do not change the feature representation; the inference process does not yield a different outcome.

Decision Model	MRR
Baseline model, used in (KR'06,GR'08)	
Model 2	0.51
Models presented in this paper	
Model 1	0.713
Model 3	0.715
Model 4	0.832
Model 5	0.848

Table 1: Results of the different transliteration models, trained using 250 samples. To facilitate readability (Klementiev and Roth, 2006b; Goldwasser and Roth, 2008) are referenced as KR'06 and GR'08 respectively.

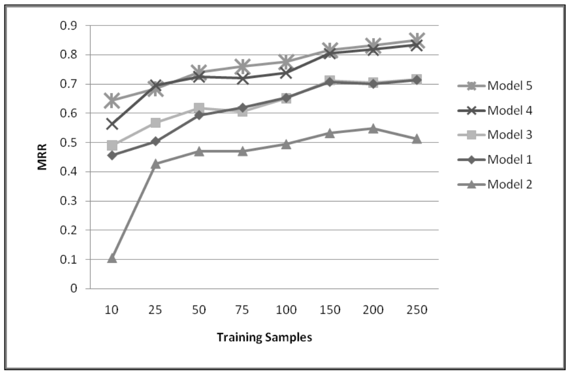


Figure 6: Results of the different constraint optimization transliteration models. Performance is compared relative to the number of samples used for training.

3.1 Normalized Objective Function

Formulating the transliteration decision as an optimization problem also allows us to naturally encode other considerations into our objective function. in this case we give preference to matching short words. We encode this preference as a normalization factor for the objective function. When evaluating on pair (w_s, w_t), we divide the weight vector length of the shorter word; our decision model now becomes:

Decision Model 6 (Model 5 - LengthNormalization)

Decision Model	MRR
Model 5	0.848
Model 5 - LN	0.894

Table 2: Results of using model 5 with and without a normalized objective function. Both models were trained using 250 samples. The LN suffix in the model's name indicate that the objective function used length normalization.

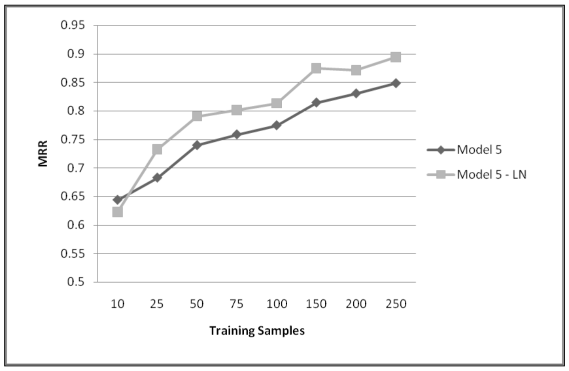


Figure 7: Results of using model 5 with and without a normalized objective function. Performance is compared relative to the number of samples used for training.

Ranking the transliteration candidates is done by evaluating:

$$s^* = argmax_i \ W_D \cdot IF_{W_D}(s_i)/min(|w_s|, |w_t|)$$

As described in table 2 and figure 7, using length normalization significantly improves the results. This can be attributed to the fact that typically Hebrew names are shorter and therefore every pair (w_s, w_t) considered by our model will be effected differently by this normalization factor.

4 Discussion

We introduced a new approach for identifying NE transliteration, viewing the transliteration decision as a global inference problem. We explored several methods for combining discriminative learning in a global constraint optimization framework and showed that discriminatively learning the objective function improves performance significantly.

From an algorithmic perspective, our key contribution is the introduction of a new method, in which learning and inference are used in an integrated way. We use learning to generate an objective function for the inference process; use the inference process to generate a better representation for the learning process, and iterate these stages.

From the transliteration perspective, our key contribution is in deriving and showing the significance of a good representation for a pair of NEs. Our representation captures both phonetic similarity and distinctive occurrence patterns across character level matchings of the two input strings, while enforcing the constraints induced by the interdependencies of the individual matchings. As we show, this representation serves to improve the ability of a discriminative learning algorithm to weigh features appropriately and results in significantly better transliteration models. This representation can be viewed as a compromise between models that do not consider dependencies between local decisions and those that try to align the two strings. Achieving this compromise is one of the advantages of the flexibility allowed by the constrained optimization framework we use. We plan to investigate using more constraints within this framework, such as soft constraints which can penalize unlikely local decisions while not completely eliminating the entire solution.

Acknowledgments

We wish to thank Alex Klementiev and the anonymous reviewers for their insightful comments. This work is partly supported by NSF grant SoD-HCER-0613885 and DARPA funding under the Bootstrap Learning Program.

References

R. Barzilay and M. Lapata. 2006. Aggregation via Set Partitioning for Natural Language Generation. In *Proceedings of HLT/NAACL*, pages 359–366, New York City, USA, June. Association for Computational Linguistics.

S. Bergsma and G. Kondrak. 2007. Alignment-based discriminative string similarity. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 656–663, Prague, Czech Republic, June. Association for Computational Linguistics.

J. Clarke and M. Lapata. Modeling compression with discourse constraints. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Lan-*

guage Processing and Computational Natural Language Learning (EMNLP-CoNLL), pages 1–11.

A. R. Golding and D. Roth. 1999. A Winnow based approach to context-sensitive spelling correction. *Machine Learning*, 34(1-3):107–130.

D. Goldwasser and D. Roth. 2008. Active sample selection for named entity transliteration. In *Proceedings of ACL-08: HLT, Short Papers*, Columbus, OH, USA, Jun. Association for Computational Linguistics.

A. Grove and D. Roth. 2001. Linear concepts and hidden variables. *Machine Learning*, 42(1/2):123–141.

Ulf Hermjakob, Kevin Knight, and Hal Daumé III. 2008. Name translation in statistical machine translation - learning when to transliterate. In *Proceedings of ACL-08: HLT*, pages 389–397, Columbus, Ohio, June. Association for Computational Linguistics.

A. Klementiev and D. Roth. 2006a. Named entity transliteration and discovery from multilingual comparable corpora. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 82–88, June.

A. Klementiev and D. Roth. 2006b. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Proc. of the Annual Meeting of the ACL*, July.

T. Marciniak and M. Strube. 2005. Beyond the Pipeline: Discrete Optimization in NLP. In *Proceedings of the Ninth CoNLL*, pages 136–143, Ann Arbor, Michigan, June. Association for Computational Linguistics.

A. Y. Ng and M. I. Jordan. 2001. On discriminative vs. generative classifiers: A comparison of logistic regression and naïve bayes. In *Neural Information Processing Systems*, pages 841–848.

V. Punyakanok, D. Roth, and W. Yih. 2005. The necessity of syntactic parsing for semantic role labeling. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1117–1123.

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In Hwee Tou Ng and Ellen Riloff, editors, *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages 1–8. Association for Computational Linguistics.

D. Roth. 1998. Learning to resolve natural language ambiguities: A unified approach. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 806–813.

Tao Tao, Su-Youn Yoon, Andrew Fister, Richard Sproat, and ChengXiang Zhai. 2006. Unsupervised named entity transliteration using temporal and phonetic correlation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 250–257, Sydney, Australia, July. Association for Computational Linguistics.

Studying the History of Ideas Using Topic Models

David Hall
Symbolic Systems
Stanford University
Stanford, CA 94305, USA
dlwh@stanford.edu

Daniel Jurafsky
Linguistics
Stanford University
Stanford, CA 94305, USA
jurafsky@stanford.edu

Christopher D. Manning
Computer Science
Stanford University
Stanford, CA 94305, USA
manning@stanford.edu

Abstract

How can the development of ideas in a scientific field be studied over time? We apply unsupervised topic modeling to the ACL Anthology to analyze historical trends in the field of Computational Linguistics from 1978 to 2006. We induce topic clusters using Latent Dirichlet Allocation, and examine the strength of each topic over time. Our methods find trends in the field including the rise of probabilistic methods starting in 1988, a steady increase in applications, and a sharp decline of research in semantics and understanding between 1978 and 2001, possibly rising again after 2001. We also introduce a model of the diversity of ideas, *topic entropy*, using it to show that COLING is a more diverse conference than ACL, but that both conferences as well as EMNLP are becoming broader over time. Finally, we apply Jensen-Shannon divergence of topic distributions to show that all three conferences are converging in the topics they cover.

1 Introduction

How can we identify and study the exploration of ideas in a scientific field over time, noting periods of gradual development, major ruptures, and the waxing and waning of both topic areas and connections with applied topics and nearby fields? One important method is to make use of citation graphs (Garfield, 1955). This enables the use of graph-based algorithms like PageRank for determining researcher or paper centrality, and examining whether their influence grows or diminishes over time.

However, because we are particularly interested in the change of *ideas* in a field over time, we have chosen a different method, following Kuhn (1962). In Kuhn's model of scientific change, science proceeds by shifting from one paradigm to another. Because researchers' ideas and vocabulary are constrained by their paradigm, successive incommensurate paradigms will naturally have different vocabulary and framing.

Kuhn's model is intended to apply only to very large shifts in scientific thought rather than at the micro level of trends in research foci. Nonetheless, we propose to apply Kuhn's insight that vocabulary and vocabulary shift is a crucial indicator of ideas and shifts in ideas. Our operationalization of this insight is based on the unsupervised topic model Latent Dirichlet Allocation (LDA; Blei et al. (2003)).

For many fields, doing this kind of historical study would be very difficult. Computational linguistics has an advantage, however: the ACL Anthology, a public repository of all papers in the *Computational Linguistics* journal and the conferences and workshops associated with the ACL, COLING, EMNLP, and so on. The ACL Anthology (Bird, 2008), and comprises over 14,000 documents from conferences and the journal, beginning as early as 1965 through 2008, indexed by conference and year. This resource has already been the basis of citation analysis work, for example, in the ACL Anthology Network of Joseph and Radev (2007). We apply LDA to the text of the papers in the ACL Anthology to induce topics, and use the trends in these topics over time and over conference venues to address questions about the development of the field.

Venue	# Papers	Years	Frequency
Journal	1291	1974–Present	Quarterly
ACL	2037	1979-Present	Yearly
EACL	596	1983–Present	∼2 Years
NAACL	293	2000–Present	∼Yearly
Applied NLP	346	1983–2000	∼3 Years
COLING	2092	1965-Present	2 Years
HLT	957	1986–Present	∼2 Years
Workshops	2756	1990-Present	Yearly
TINLAP	128	1975–1987	Rarely
MUC	160	1991–1998	∼2 Years
IJCNLP	143	2005	——
Other	120	——	——

Table 1: Data in the ACL Anthology

Despite the relative youth of our field, computational linguistics has witnessed a number of research trends and shifts in focus. While some trends are obvious (such as the rise in machine learning methods), others may be more subtle. Has the field gotten more theoretical over the years or has there been an increase in applications? What topics have declined over the years, and which ones have remained roughly constant? How have fields like Dialogue or Machine Translation changed over the years? Are there differences among the conferences, for example between COLING and ACL, in their interests and breadth of focus? As our field matures, it is important to go beyond anecdotal description to give grounded answers to these questions. Such answers could also help give formal metrics to model the differences between the many conferences and venues in our field, which could influence how we think about reviewing, about choosing conference topics, and about long range planning in our field.

2 Methodology

2.1 Data

The analyses in this paper are based on a text-only version of the Anthology that comprises some 12,500 papers. The distribution of the Anthology data is shown in Table 1.

2.2 Topic Modeling

Our experiments employ Latent Dirichlet Allocation (LDA; Blei et al. (2003)), a generative latent variable model that treats documents as bags of words generated by one or more topics. Each document is characterized by a multinomial distribution over topics, and each topic is in turn characterized by a multinomial distribution over words. We perform parameter estimation using collapsed Gibbs sampling (Griffiths and Steyvers, 2004).

Possible extensions to this model would be to integrate topic modelling with citations (e.g., Dietz et al. (2007), Mann et al. (2006), and Jo et al. (2007)). Another option is the use of more fine-grained or hierarchical model (e.g., Blei et al. (2004), and Li and McCallum (2006)).

All our studies measure change in various aspects of the ACL Anthology over time. LDA, however, does not explicitly model temporal relationships. One way to model temporal relationships is to employ an extension to LDA. The Dynamic Topic Model (Blei and Lafferty, 2006), for example, represents each year's documents as generated from a normal distribution centroid over topics, with the following year's centroid generated from the preceding year's. The Topics over Time Model (Wang and McCallum, 2006) assumes that each document chooses its own time stamp based on a topic-specific beta distribution.

Both of these models, however, impose constraints on the time periods. The Dynamic Topic Model penalizes large changes from year to year while the beta distributions in Topics over Time are relatively inflexible. We chose instead to perform *post hoc* calculations based on the observed probability of each topic given the current year. We define $\hat{p}(z|y)$ as the empirical probability that an arbitrary paper d written in year y was about topic z:

$$\begin{aligned}\hat{p}(z|y) &= \sum_{d:t_d=y} \hat{p}(z|d)\hat{p}(d|y) \\ &= \frac{1}{C}\sum_{d:t_d=y} \hat{p}(z|d) \\ &= \frac{1}{C}\sum_{d:t_d=y}\sum_{z'_i\in d} I(z'_i = z)\end{aligned} \quad (1)$$

where I is the indicator function, t_d is the date document d was written, $\hat{p}(d|y)$ is set to a constant $1/C$.

3 Summary of Topics

We first ran LDA with 100 topics, and took 36 that we found to be relevant. We then hand-selected seed

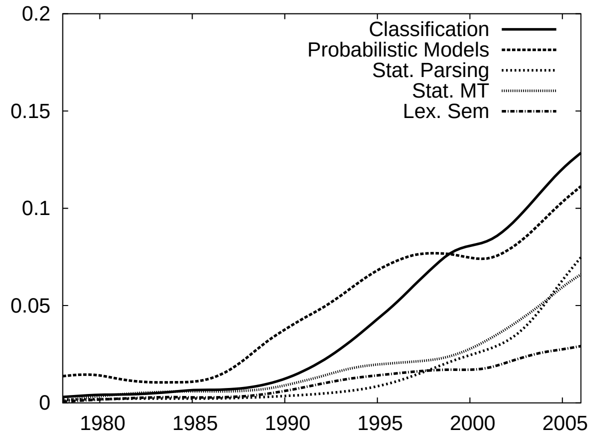


Figure 1: Topics in the ACL Anthology that show a strong recent increase in strength.

words for 10 more topics to improve coverage of the field. These 46 topics were then used as priors to a new 100-topic run. The top ten most frequent words for 43 of the topics along with hand-assigned labels are listed in Table 2. Topics deriving from manual seeds are marked with an asterisk.

4 Historical Trends in Computational Linguistics

Given the space of possible topics defined in the previous section, we now examine the history of these in the entire ACL Anthology from 1978 until 2006. To visualize some trends, we show the probability mass associated with various topics over time, plotted as (a smoothed version of) $\hat{p}(z|y)$.

4.1 Topics Becoming More Prominent

Figure 1 shows topics that have become more prominent more recently.

Of these new topics, the rise in *probabilistic models* and *classification/tagging* is unsurprising. In order to distinguish these two topics, we show 20 of the strongly weighted words:

Probabilistic Models: *model word probability set data number algorithm language corpus method figure probabilities table test statistical distribution function al values performance*

Classification/Tagging: *features data corpus set feature table word tag al test accuracy pos classification performance tags tagging text task information class*

Some of the papers with the highest weights for the *probabilistic models* class include:

N04-1039	Goodman, Joshua. Exponential Priors For Maximum Entropy Models (HLT-NAACL, 2004)
W97-0309	Saul, Lawrence, Pereira, Fernando C. N. Aggregate And Mixed-Order Markov Models For Statistical Language Processing (EMNLP, 1997)
P96-1041	Chen, Stanley F., Goodman, Joshua. An Empirical Study Of Smoothing Techniques For Language Modeling (ACL, 1996)
H89-2013	Church, Kenneth Ward, Gale, William A. Enhanced Good-Turing And CatCal: Two New Methods For Estimating Probabilities Of English Bigrams (Workshop On Speech And Natural Language, 1989)
P02-1023	Gao, Jianfeng, Zhang, Min Improving Language Model Size Reduction Using Better Pruning Criteria (ACL, 2002)
P94-1038	Dagan, Ido, Pereira, Fernando C. N. Similarity-Based Estimation Of Word Cooccurrence Probabilities (ACL, 1994)

Some of the papers with the highest weights for the *classification/tagging* class include:

W00-0713	Van Den Bosch, Antal Using Induced Rules As Complex Features In Memory-Based Language Learning (CoNLL, 2000)
W01-0709	Estabrooks, Andrew, Japkowicz, Nathalie A Mixture-Of-Experts Framework For Text Classification (Workshop On Computational Natural Language Learning CoNLL, 2001)
A00-2035	Mikheev, Andrei. Tagging Sentence Boundaries (ANLP-NAACL, 2000)
H92-1022	Brill, Eric. A Simple Rule-Based Part Of Speech Tagger (Workshop On Speech And Natural Language, 1992)

As Figure 1 shows, *probabilistic models* seem to have arrived significantly before *classifiers*. The probabilistic model topic increases around 1988, which seems to have been an important year for probabilistic models, including high-impact papers like A88-1019 and C88-1016 below. The ten papers from 1988 with the highest weights for the *probabilistic model* and *classifier* topics were the following:

C88-1071	Kuhn, Roland. Speech Recognition and the Frequency of Recently Used Words (COLING)
J88-1003	DeRose, Steven. Grammatical Category Disambiguation by Statistical Optimization. (CL Journal)
C88-2133	Su, Keh-Yi, and Chang, Jing-Shin. Semantic and Syntactic Aspects of Score Function. (COLING)
A88-1019	Church, Kenneth Ward. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. (ANLP)
C88-2134	Sukhotin, B.V. Optimization Algorithms of Deciphering as the Elements of a Linguistic Theory. (COLING)
P88-1013	Haigh, Robin, Sampson, Geoffrey, and Atwell, Eric. Project APRIL: a progress report. (ACL)
A88-1005	Boggess, Lois. Two Simple Prediction Algorithms to Facilitate Text Production. (ANLP)
C88-1016	Peter F. Brown, et al. A Statistical Approach to Machine Translation. (COLING)
A88-1028	Oshika, Beatrice, et al.. Computational Techniques for Improved Name Search. (ANLP)
C88-1020	Campbell, W.N. Speech-rate Variation and the Prediction of Duration. (COLING)

What do these early papers tell us about how

Anaphora Resolution	resolution anaphora pronoun discourse antecedent pronouns coreference reference definite algorithm
Automata	string state set finite context rule algorithm strings language symbol
Biomedical	medical protein gene biomedical wkh abstracts medline patient clinical biological
Call Routing	call caller routing calls destination vietnamese routed router destinations gorin
Categorial Grammar	proof formula graph logic calculus axioms axiom theorem proofs lambek
Centering*	centering cb discourse cf utterance center utterances theory coherence entities local
Classical MT	japanese method case sentence analysis english dictionary figure japan word
Classification/Tagging	features data corpus set feature table word tag al test
Comp. Phonology	vowel phonological syllable phoneme stress phonetic phonology pronunciation vowels phonemes
Comp. Semantics*	semantic logical semantics john sentence interpretation scope logic form set
Dialogue Systems	user dialogue system speech information task spoken human utterance language
Discourse Relations	discourse text structure relations rhetorical relation units coherence texts rst
Discourse Segment.	segment segmentation segments chain chains boundaries boundary seg cohesion lexical
Events/Temporal	event temporal time events tense state aspect reference relations relation
French Function	de le des les en une est du par pour
Generation	generation text system language information knowledge natural figure domain input
Genre Detection	genre stylistic style genres fiction humor register biber authorship registers
Info. Extraction	system text information muc extraction template names patterns pattern domain
Information Retrieval	document documents query retrieval question information answer term text web
Lexical Semantics	semantic relations domain noun corpus relation nouns lexical ontology patterns
MUC Terrorism	slot incident tgt target id hum phys type fills perp
Metaphor	metaphor literal metonymy metaphors metaphorical essay metonymic essays qualia analogy
Morphology	word morphological lexicon form dictionary analysis morphology lexical stem arabic
Named Entities*	entity named entities ne names ner recognition ace nes mentions mention
Paraphrase/RTE	paraphrases paraphrase entailment paraphrasing textual para rte pascal entailed dagan
Parsing	parsing grammar parser parse rule sentence input left grammars np
Plan-Based Dialogue	plan discourse speaker action model goal act utterance user information
Probabilistic Models	model word probability set data number algorithm language corpus method
Prosody	prosodic speech pitch boundary prosody phrase boundaries accent repairs intonation
Semantic Roles*	semantic verb frame argument verbs role roles predicate arguments
Yale School Semantics	knowledge system semantic language concept representation information network concepts base
Sentiment	subjective opinion sentiment negative polarity positive wiebe reviews sentence opinions
Speech Recognition	speech recognition word system language data speaker error test spoken
Spell Correction	errors error correction spelling ocr correct corrections checker basque corrected detection
Statistical MT	english word alignment language source target sentence machine bilingual mt
Statistical Parsing	dependency parsing treebank parser tree parse head model al np
Summarization	sentence text evaluation document topic summary summarization human summaries score
Syntactic Structure	verb noun syntactic sentence phrase np subject structure case clause
TAG Grammars*	tree node trees nodes derivation tag root figure adjoining grammar
Unification	feature structure grammar lexical constraints unification constraint type structures rule
WSD*	word senses wordnet disambiguation lexical semantic context similarity dictionary
Word Segmentation	chinese word character segmentation corpus dictionary korean language table system
WordNet*	synset wordnet synsets hypernym ili wordnets hypernyms eurowordnet hyponym ewn wn

Table 2: Top 10 words for 43 of the topics. Starred topics are hand-seeded.

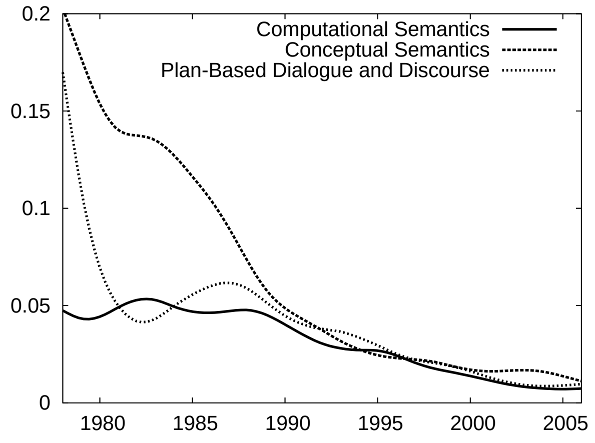


Figure 2: Topics in the ACL Anthology that show a strong decline from 1978 to 2006.

probabilistic models and classifiers entered the field? First, not surprisingly, we note that the vast majority (9 of 10) of the papers appeared in conference proceedings rather than the journal, confirming that in general new ideas appear in conferences. Second, of the 9 conference papers, most of them appeared in the COLING conference (5) or the ANLP workshop (3) compared to only 1 in the ACL conference. This suggests that COLING may have been more receptive than ACL to new ideas at the time, a point we return to in Section 6. Finally, we examined the background of the authors of these papers. Six of the 10 papers either focus on speech (C88-1010, A88-1028, C88-1071) or were written by authors who had previously published on speech recognition topics, including the influential IBM (Brown *et al.*) and AT&T (Church) labs (C88-1016, A88-1005, A88-1019). Speech recognition is historically an electrical engineering field which made quite early use of probabilistic and statistical methodologies. This suggests that researchers working on spoken language processing were an important conduit for the borrowing of statistical methodologies into computational linguistics.

4.2 Topics That Have Declined

Figure 2 shows several topics that were more prominent at the beginning of the ACL but which have shown the most precipitous decline. Papers strongly associated with the *plan-based dialogue* topic include:

J99-1001	Carberry, Sandra, Lambert, Lynn. A Process Model For Recognizing Communicative Acts And Modeling Negotiation Subdialogues (CL, 1999)
J95-4001	McRoy, Susan W., Hirst, Graeme. The Repair Of Speech Act Misunderstandings By Abductive Inference (CL, 1995)
P93-1039	Chu, Jennifer. Responding To User Queries In A Collaborative Environment (ACL, 1993)
P86-1032	Pollack, Martha E. A Model Of Plan Inference That Distinguishes Between The Beliefs Of Actors And Observers (ACL, 1986)
T78-1017	Perrault, Raymond C., Allen, James F. Speech Acts As A Basis For Understanding Dialogue Coherence (Theoretical Issues In Natural Language Processing, 1978)
P84-1063	Litman, Diane J., Allen, James F. A Plan Recognition Model For Clarification Subdialogues (COLING-ACL, 1984)

Papers strongly associated with the *computational semantics* topic include:

J90-4002	Haas, Andrew R. Sentential Semantics For Propositional Attitudes (CL, 1990)
P83-1009	Hobbs, Jerry R. An Improper Treatment Of Quantification In Ordinary English (ACL, 1983)
J87-1005	Hobbs, Jerry R., Shieber, Stuart M. An Algorithm For Generating Quantifier Scopings (CL, 1987)
C90-1003	Johnson, Mark, Kay, Martin. Semantic Abstraction And Anaphora (COLING, 1990)
P89-1004	Alshawi, Hiyan, Van Eijck, Jan. Logical Forms In The Core Language Engine (ACL, 1989)

Papers strongly associated with the *conceptual semantics/story understanding* topic include:

C80-1022	Ogawa, Hitoshi, Nishi, Junichiro, Tanaka, Kokichi. The Knowledge Representation For A Story Understanding And Simulation System (COLING, 1980)
A83-1012	Pazzani, Michael J., Engelman, Carl. Knowledge Based Question Answering (ANLP, 1983)
P82-1029	McCoy, Kathleen F. Augmenting A Database Knowledge Representation For Natural Language Generation (ACL, 1982)
H86-1010	Ksiezyk, Tomasz, Grishman, Ralph An Equipment Model And Its Role In The Interpretation Of Nominal Compounds (Workshop On Strategic Computing - Natural Language, 1986)
P80-1030	Wilensky, Robert, Arens, Yigal. PHRAN - A Knowledge-Based Natural Language Understander (ACL, 1980)
A83-1013	Boguraev, Branimir K., Sparck Jones, Karen. How To Drive A Database Front End Using General Semantic Information (ANLP, 1983)
P79-1003	Small, Steven L. Word Expert Parsing (ACL, 1979)

The declines in both computational semantics and *conceptual semantics/story understanding* suggests that it is possible that the entire field of natural language understanding and computational semantics broadly construed has fallen out of favor. To see if this was in fact the case we created a metatopic called *semantics* in which we combined various semantics topics (not including pragmatic topics like anaphora resolution or discourse coherence) including: *lexical semantics, conceptual semantics/story*

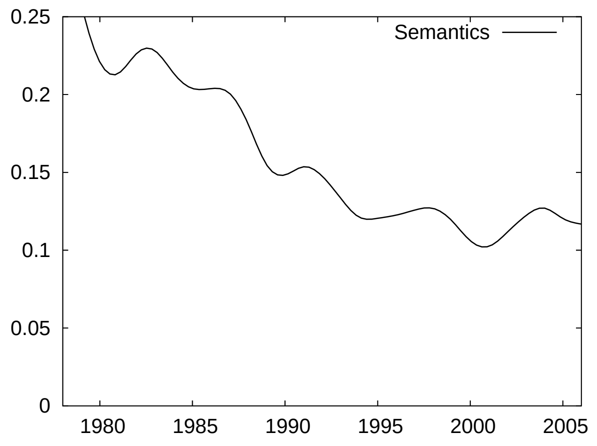


Figure 3: Semantics over time

understanding, computational semantics, WordNet, word sense disambiguation, semantic role labeling, RTE and paraphrase, MUC information extraction, and *events/temporal*. We then plotted $\hat{p}(z \in S|y)$, the sum of the proportions per year for these topics, as shown in Figure 3. The steep decrease in semantics is readily apparent. The last few years has shown a levelling off of the decline, and possibly a revival of this topic; this possibility will need to be confirmed as we add data from 2007 and 2008.

We next chose two fields, Dialogue and Machine Translation, in which it seemed to us that the topics discovered by LDA suggested a shift in paradigms in these fields. Figure 4 shows the shift in translation, while Figure 5 shows the change in dialogue.

The shift toward statistical machine translation is well known, at least anecdotally. The shift in dialogue seems to be a move toward more applied, speech-oriented, or commercial dialogue systems and away from more theoretical models.

Finally, Figure 6 shows the history of several topics that peaked at intermediate points throughout the history of the field. We can see the peak of *unification* around 1990, of *syntactic structure* around 1985 of *automata* in 1985 and again in 1997, and of *word sense disambiguation* around 1998.

5 Is Computational Linguistics Becoming More Applied?

We don't know whether our field is becoming more applied, or whether perhaps there is a trend towards new but unapplied theories. We therefore

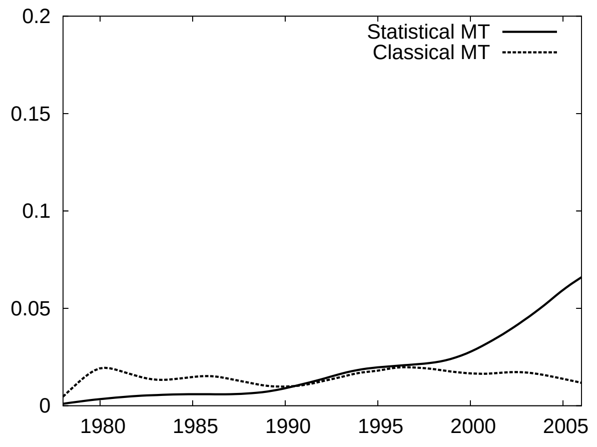


Figure 4: Translation over time

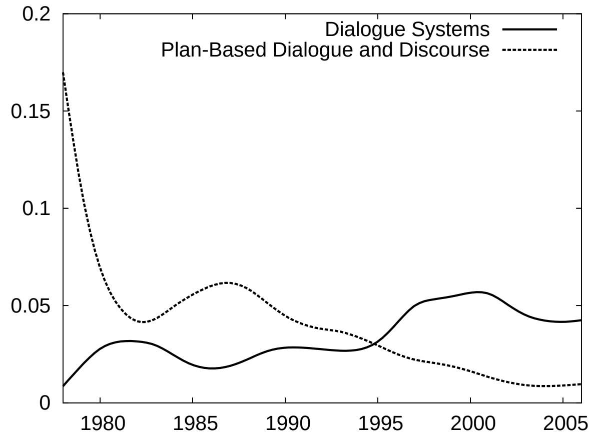


Figure 5: Dialogue over time

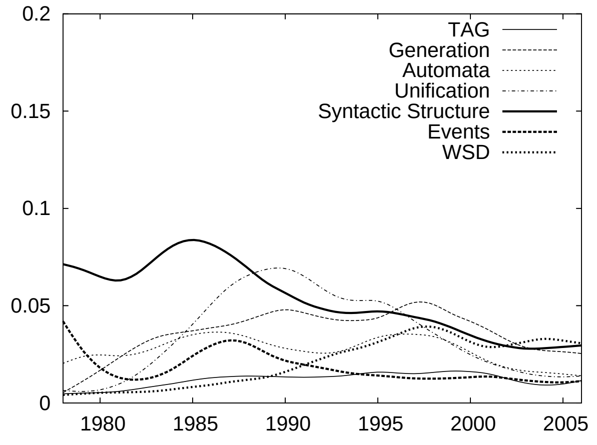


Figure 6: Peaked topics

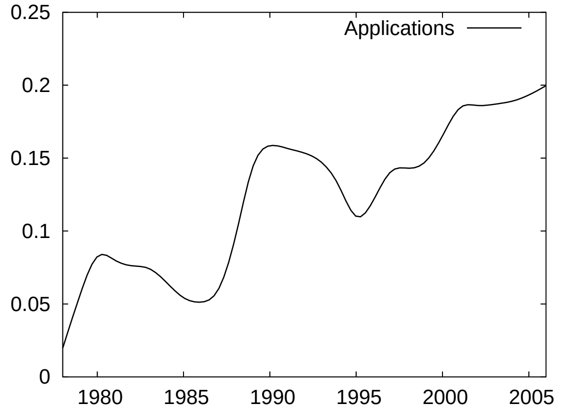


Figure 7: Applications over time

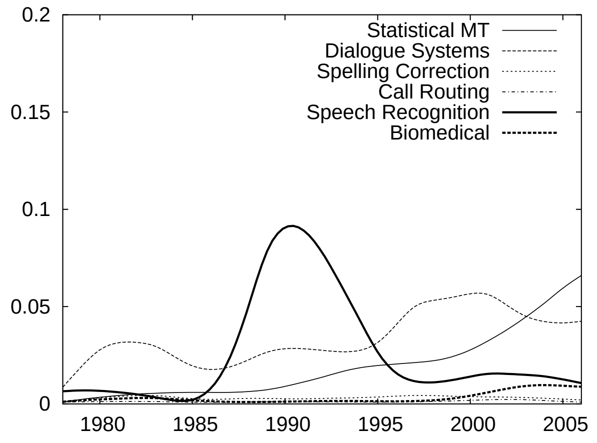


Figure 8: Six applied topics over time

looked at trends over time for the following applications: *Machine Translation*, *Spelling Correction*, *Dialogue Systems*, *Information Retrieval*, *Call Routing*, *Speech Recognition*, and *Biomedical* applications.

Figure 7 shows a clear trend toward an increase in applications over time. The figure also shows an interesting bump near 1990. Why was there such a sharp temporary increase in applications at that time? Figure 8 shows details for each application, making it clear that the bump is caused by a temporary spike in the *Speech Recognition* topic.

In order to understand why we see this temporary spike, Figure 9 shows the unsmoothed values of the *Speech Recognition* topic prominence over time.

Figure 9 clearly shows a huge spike for the years 1989–1994. These years correspond exactly to the DARPA Speech and Natural Language Workshop,

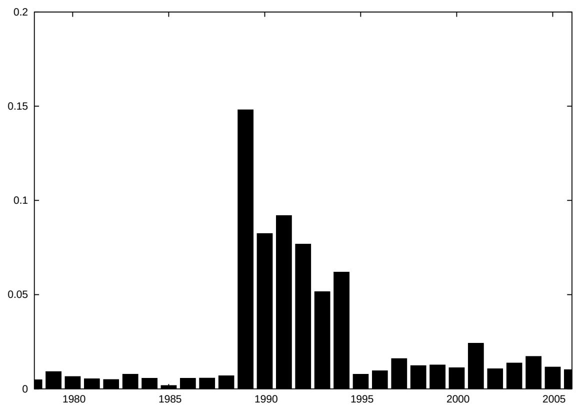


Figure 9: Speech recognition over time

held at different locations from 1989–1994. That workshop contained a significant amount of speech until its last year (1994), and then it was revived in 2001 as the Human Language Technology workshop with a much smaller emphasis on speech processing. It is clear from Figure 9 that there is still some speech research appearing in the Anthology after 1995, certainly more than the period before 1989, but it's equally clear that speech recognition is not an application that the ACL community has been successful at attracting.

6 Differences and Similarities Among COLING, ACL, and EMNLP

The computational linguistics community has two distinct conferences, COLING and ACL, with different histories, organizing bodies, and philosophies. Traditionally, COLING was larger, with parallel sessions and presumably a wide variety of topics, while ACL had single sessions and a more narrow scope. In recent years, however, ACL has moved to parallel sessions, and the conferences are of similar size. Has the distinction in breadth of topics also been blurred? What are the differences and similarities in topics and trends between these two conferences?

More recently, the EMNLP conference grew out of the Workshop on Very Large Corpora, sponsored by the Special Interest Group on Linguistic Data and corpus-based approaches to NLP (SIGDAT). EMNLP started as a much smaller and narrower conference but more recently, while still smaller than both COLING and ACL, it has grown large enough to be considered with them. How does the breadth of its topics compare with the others?

Our hypothesis, based on our intuitions as conference attendees, is that ACL is still more narrow in scope than COLING, but has broadened considerably. Similarly, our hypothesis is that EMNLP has begun to broaden considerably as well, although not to the extent of the other two.

In addition, we're interested in whether the topics of these conferences are converging or not. Are the probabilistic and machine learning trends that are dominant in ACL becoming dominant in COLING as well? Is EMNLP adopting some of the topics that are popular at COLING?

To investigate both of these questions, we need a model of the topic distribution for each conference. We define the empirical distribution of a topic z at a conference c, denoted by $\hat{p}(z|c)$ by:

$$\begin{aligned}\hat{p}(z|c) &= \sum_{d:c_d=c} \hat{p}(z|d)\hat{p}(d|c) \\ &= \frac{1}{C}\sum_{d:c_d=c} \hat{p}(z|d) \\ &= \frac{1}{C}\sum_{d:c_d=c}\sum_{z_i'\in d} I(z_i' = z)\end{aligned} \tag{2}$$

We also condition on the year for each conference, giving us $\hat{p}(z|y,c)$.

We propose to measure the breadth of a conference by using what we call *topic entropy*: the conditional entropy of this conference topic distribution. Entropy measures the average amount of information expressed by each assignment to a random variable. If a conference has higher topic entropy, then it more evenly divides its probability mass across the generated topics. If it has lower, it has a far more narrow focus on just a couple of topics. We therefore measured *topic entropy*:

$$H(z|c,y) = -\sum_{i=1}^{K} \hat{p}(z_i|c,y)\log \hat{p}(z_i|c,y) \tag{3}$$

Figure 10 shows the conditional topic entropy of each conference over time. We removed from the ACL and COLING lines the years when ACL and COLING are colocated (1984, 1998, 2006), and marked those colocated years as points separate from either plot. As expected, COLING has been historically the broadest of the three conferences, though perhaps slightly less so in recent years. ACL started with a fairly narrow focus, but became nearly as broad as COLING during the 1990's. However, in the past 8 years it has become more narrow again, with a steeper decline in breadth than COLING. EMNLP, true to its status as a "Special Interest" conference, began as a very narrowly focused conference, but now it seems to be catching up to at least ACL in terms of the breadth of its focus.

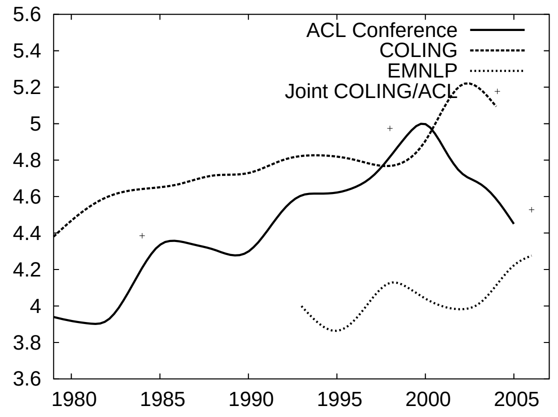


Figure 10: Entropy of the three major conferences per year

Since the three major conferences seem to be converging in terms of breadth, we investigated whether or not the topic distributions of the conferences were also converging. To do this, we plotted the Jensen-Shannon (JS) divergence between each pair of conferences. The Jensen-Shannon divergence is a symmetric measure of the similarity of two pairs of distributions. The measure is 0 only for identical distributions and approaches infinity as the two differ more and more. Formally, it is defined as the average of the KL divergence of each distribution to the average of the two distributions:

$$\begin{aligned}\mathrm{D}_{\mathrm{JS}}(P||Q) &= \frac{1}{2}\mathrm{D}_{\mathrm{KL}}(P||R) + \frac{1}{2}\mathrm{D}_{\mathrm{KL}}(Q||R) \\ R &= \frac{1}{2}(P+Q)\end{aligned} \tag{4}$$

Figure 11 shows the JS divergence between each pair of conferences over time. Note that EMNLP

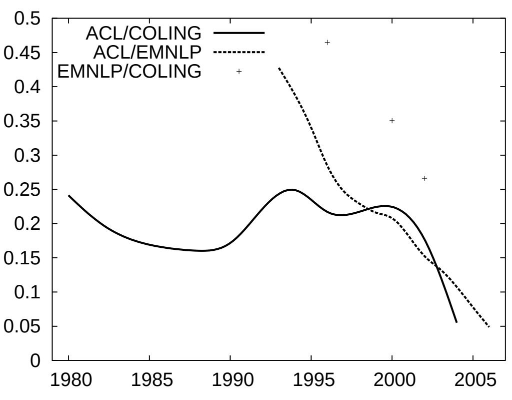


Figure 11: JS Divergence between the three major conferences

and COLING have historically met very infrequently in the same year, so those similarity scores are plotted as points and not smoothed. The trend across all three conferences is clear: each conference is not only increasing in breadth, but also in similarity. In particular, EMNLP and ACL's differences, once significant, are nearly erased.

7 Conclusion

Our method discovers a number of trends in the field, such as the general increase in applications, the steady decline in semantics, and its possible reversal. We also showed a convergence over time in topic coverage of ACL, COLING, and EMNLP as well an expansion of topic diversity. This growth and convergence of the three conferences, perhaps influenced by the need to increase recall (Church, 2005) seems to be leading toward a tripartite realization of a single new "latent" conference.

Acknowledgments

Many thanks to Bryan Gibson and Dragomir Radev for providing us with the data behind the ACL Anthology Network. Also to Sharon Goldwater and the other members of the Stanford NLP Group as well as project Mimir for helpful advice. Finally, many thanks to the Office of the President, Stanford University, for partial funding.

References

Steven Bird. 2008. Association of Computational Linguists Anthology. http://www.aclweb.org/anthology-index/.

David Blei and John D. Lafferty. 2006. Dynamic topic models. *ICML*.

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

D. Blei, T. Gri, M. Jordan, and J. Tenenbaum. 2004. Hierarchical topic models and the nested Chinese restaurant process.

Kenneth Church. 2005. Reviewing the reviewers. *Comput. Linguist.*, 31(4):575–578.

Laura Dietz, Steffen Bickel, and Tobias Scheffer. 2007. Unsupervised prediction of citation influences. In *ICML*, pages 233–240. ACM.

Eugene Garfield. 1955. Citation indexes to science: A new dimension in documentation through association of ideas. *Science*, 122:108–111.

Tom L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *PNAS*, 101 Suppl 1:5228–5235, April.

Yookyung Jo, Carl Lagoze, and C. Lee Giles. 2007. Detecting research topics via the correlation between graphs and texts. In *KDD*, pages 370–379, New York, NY, USA. ACM.

Mark T. Joseph and Dragomir R. Radev. 2007. Citation analysis, centrality, and the ACL anthology. Technical Report CSE-TR-535-07, University of Michigan. Department of Electrical Engineering and Computer Science.

Thomas S. Kuhn. 1962. *The Structure of Scientific Revolutions*. University Of Chicago Press.

Wei Li and Andrew McCallum. 2006. Pachinko allocation: DAG-structured mixture models of topic correlations. In *ICML*, pages 577–584, New York, NY, USA. ACM.

Gideon S. Mann, David Mimno, and Andrew McCallum. 2006. Bibliometric impact measures leveraging topic analysis. In *JCDL '06: Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries*, pages 65–74, New York, NY, USA. ACM.

Xuerui Wang and Andrew McCallum. 2006. Topics over time: a non-Markov continuous-time model of topical trends. In *KDD*, pages 424–433, New York, NY, USA. ACM.

Triplet Lexicon Models for Statistical Machine Translation

Saša Hasan, Juri Ganitkevitch, Hermann Ney, Jesús Andrés-Ferrer†[*]
Human Language Technology and Pattern Recognition, RWTH Aachen University, Germany
†Universidad Politécnica de Valencia, Dept. Sist. Informáticos y Computación
{hasan,ganitkevitch,ney}@cs.rwth-aachen.de jandres@dsic.upv.es

Abstract

This paper describes a lexical trigger model for statistical machine translation. We present various methods using triplets incorporating long-distance dependencies that can go beyond the local context of phrases or n-gram based language models. We evaluate the presented methods on two translation tasks in a reranking framework and compare it to the related IBM model 1. We show slightly improved translation quality in terms of BLEU and TER and address various constraints to speed up the training based on Expectation-Maximization and to lower the overall number of triplets without loss in translation performance.

1 Introduction

Data-driven methods have been applied very successfully within the machine translation domain since the early 90s. Starting from single-word-based translation approaches, significant improvements have been made through advances in modeling, availability of larger corpora and more powerful computers. Thus, substantial progress made in the past enables today's MT systems to achieve acceptable results in terms of translation quality for specific language pairs such as Arabic-English. If sufficient amounts of parallel data are available, statistical MT systems can be trained on millions of

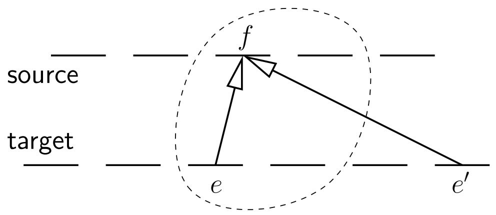


Figure 1: Triplet example: a source word f is triggered by two target words e and e', where one of the words is within and the other outside the considered phrase pair (indicated by the dashed line).

sentence pairs and use an extended level of context based on bilingual groups of words which denote the building blocks of state-of-the-art phrase-based SMT systems.

Due to data sparseness, statistical models are often trained on local context only. Language models are derived from n-grams with $n \leq 5$ and bilingual phrase pairs are extracted with lengths up to 10 words on the target side. This captures the local dependencies of the data in detail and is responsible for the success of data-driven phrase-based approaches.

In this work, we will introduce a new statistical model based on lexicalized triplets (f, e, e') which we will also refer to as cross-lingual triggers of the form $(e, e' \rightarrow f)$. This can be understood as two words in one language triggering one word in another language. These triplets, modeled by $p(f|e, e')$, are closely related to lexical translation probabilities based on the IBM model 1, i.e. $p(f|e)$. Several constraints and setups will be described later on in more detail, but as an introduction one can

[*]The work was carried out while the author was at the Human Language Technology and Pattern Recognition group at RWTH Aachen University and partly supported by the Valencian *Conselleria d'Empresa, Universitat i Ciència* under grants CTBPRA/2005/ and BEFPI/2007/014.

think of the following interpretation which is depicted in Figure 1: Using a phrase-based MT approach, a source word f is triggered by its translation e which is part of the phrase being considered, whereas another target word e' outside this phrase serves as an additional trigger in order to allow for more fine-grained distinction of a specific word sense. Thus, this cross-lingual trigger model can be seen as a combination of a lexicon model (i.e. f and e) and a model similar to monolingual long-range (i.e. distant bigram) trigger models (i.e. e and e', although these dependencies are reflected indirectly via $e' \rightarrow f$) which uses both local (in-phrase) and global (in-sentence) information for the scoring. The motivation behind this approach is to get non-local information outside the current context (i.e. the currently considered bilingual phrase pair) into the translation process. The triplets are trained via the EM algorithm, as will be shown later in more detail.

2 Related Work

In the past, a significant number of methods has been presented that try to capture long-distance dependencies, i.e. use dependencies in the data that reach beyond the local context of n-grams or phrase pairs. In language modeling, monolingual trigger approaches have been presented (Rosenfeld, 1996; Tillmann and Ney, 1997) as well as syntactical methods that parse the input and model long-range dependencies on the syntactic level by conditioning on the predecessing words and their corresponding parent nodes (Chelba and Jelinek, 2000; Roark, 2001). The latter approach was shown to reduce perplexities and improve the WER in speech recognition systems. One drawback is that the parsing process might slow down the system significantly and the approach is complicated to be integrated directly in the search process. Thus, the effect is often shown offline in reranking experiments using n-best lists.

One of the simplest models that can be seen in the context of lexical triggers is the IBM model 1 (Brown et al., 1993) which captures lexical dependencies between source and target words. It can be seen as a lexicon containing correspondents of translations of source and target words in a very broad sense since the pairs are trained on the full sentence level. The model presented in this work is very close to the initial IBM model 1 and can be seen as taking another word into the conditioning part, i.e. the triggering items.[1] Furthermore, since the second trigger can come from any part of the sentence, we also have a link to long-range monolingual triggers as presented above.

A long-range trigram model is presented in (Della Pietra et al., 1994) where it is shown how to derive a probabilistic link grammar in order to capture long-range dependencies in English using the EM algorithm. Expectation-Maximization is used in the presented triplet model as well which is described in more detail in Section 3. Instead of deriving a grammar automatically (based on POS tags of the words), we rely on a fully lexicalized approach, i.e. the training is taking place at the word level.

Related work in the context of fine-tuning language models by using cross-lingual lexical triggers is presented in (Kim and Khudanpur, 2003). The authors show how to use cross-lingual triggers on a document level in order to extract translation lexicons and domain-specific language models using a mutual information criterion.

Recently, word-sense disambiguation (WSD) methods have been shown to improve translation quality (Chan et al., 2007; Carpuat and Wu, 2007). Chan et al. (2007) use an SVM based classifier for disambiguating word senses which are directly incorporated in the decoder through additional features that are part of the log-linear combination of models. They use local collocations based on surrounding words left and right of an ambiguous word including the corresponding parts-of-speech. Although no long-range dependencies are modeled, the approach yields an improvement of +0.6% BLEU on the NIST Chinese-English task. In Carpuat and Wu (2007), another state-of-the-art WSD engine (a combination of naive Bayes, maximum entropy, boosting and Kernel PCA models) is used to dynamically determine the score of a phrase pair under consideration and, thus, let the phrase selection adapt to the context of the sentence. Although the baseline is significantly lower than in the work of Chan et al., this setup reaches an improvement of 0.5% BLEU on the NIST CE task and up to 1.1% BLEU on the

[1]Thus, instead of $p(f|e)$ we model $p(f|e, e')$ with different additional constraints as explained later on.

IWSLT'06 test sets.

The work in this paper tries to complement the WSD approaches by using long-range dependencies. If triggers from a local context determine different lexical choice for the word being triggered, the setting is comparable to the mentioned WSD approaches (although local dependencies might already be reflected sufficiently in the phrase models). A distant second trigger, however, might have a beneficial effect for specific languages, e.g. by capturing word splits (as it is the case in German for verbs with separable prefixes) or, as already mentioned, allowing for a more fine-grained lexical choice of the word being triggered, namely based on another word which is not part of the current local, i.e. phrasal, context.

The basic idea of triplets of the form $(e, f' \rightarrow f)$, called multi-word extensions, is also mentioned in (Tillmann, 2001) but neither evaluated nor investigated in further detail.

In the following sections, we will describe the model proposed in this work. In Section 3, a detailed introduction is given, as well as the EM training and variations of the model. The different settings will be evaluated in Section 4, where we show experiments on the IWSLT Chinese-English and TC-STAR EPPS English-Spanish/Spanish-English tracks. A discussion of the results and further examples are given in Section 5. Final remarks and future work are addressed in Section 6.

3 Model

As an extension to commonly used lexical word pair probabilities $p(f|e)$ as introduced in (Brown et al., 1993), we define our model to operate on word triplets. A triplet (f, e, e') is assigned a value $\alpha(f|e, e') \geq 0$ with the constraint such that

$$\forall e, e' : \sum_{f} \alpha(f|e, e') = 1.$$

Throughout this paper, e and e' will be referred to as the *first* and the *second trigger*, respectively. In view of its triggers f will be termed the *effect*.

For a given bilingual sentence pair (f_1^J, e_1^I), the probability of a source word f_j given the whole target sentence e_1^I for the triplet model is defined as:

$$p_{all}(f_j|e_1^I) = \frac{1}{Z} \sum_{i=1}^{I} \sum_{k=i+1}^{I} \alpha(f_j|e_i, e_k), \qquad (1)$$

where Z denotes a normalization factor based on the corresponding target sentence length, i.e.

$$Z = \frac{I(I-1)}{2}. \qquad (2)$$

The introduction of a second trigger (i.e. e_k in Eq. 1) enables the model to combine local (i.e. word or phrase level) and global (i.e. sentence level) information.

In the following, we will describe the training procedure of the model via maximum likelihood estimation for the unconstrained case.

3.1 Training

The goal of the training procedure is to maximize the log-likelihood F_{all} of the triplet model for a given bilingual training corpus $\{(f_1^J, e_1^I)\}_1^N$ consisting of N sentence pairs:

$$F_{all} := \sum_{n=1}^{N} \sum_{j=1}^{J_n} \log p_{all}(f_j|e_1^{I_n}),$$

where J_n and I_n are the lengths of the nth source and target sentences, respectively. As there is no closed form solution for the maximum likelihood estimate, we resort to iterative training via the EM algorithm (Dempster et al., 1977). We define the auxiliary function $Q(\mu; \bar{\mu})$ based on F_{all} where $\bar{\mu}$ is the new estimate within an iteration which is to be derived from the current estimate μ. Here, μ stands for the entire set of model parameters to be estimated, i.e. the set of all $\{\alpha(f|e, e')\}$. Thus, we obtain

$$Q\big(\{\alpha(f|e,e')\}; \{\bar{\alpha}(f|e,e')\}\big) = \sum_{n=1}^{N} \sum_{j=1}^{J_n} \sum_{i=1}^{I_n} \sum_{k=i+1}^{I_n} \left[\frac{Z_n^{-1} \alpha(f_j|e_i, e_k)}{p_{all}(f_j|e_1^{I_n})} \cdot \log\big(Z_n^{-1} \bar{\alpha}(f_j|e_i, e_k)\big) \right], \qquad (3)$$

where Z_n is defined as in Eq. 2. Using the method of Lagrangian multipliers for the normalization constraint, we take the derivative with respect to

$\bar{\alpha}(f|e,e')$ and obtain:

$$\bar{\alpha}(f|e,e') = \frac{A(f,e,e')}{\sum_{f'} A(f',e,e')} \tag{4}$$

where $A(f,e,e')$ is a relative weight accumulator over the parallel corpus:

$$A(f,e,e') = \sum_{n=1}^{N} \sum_{j=1}^{J_n} \delta(f,f_j) \frac{Z_n^{-1}\alpha(f|e,e')}{p_{all}(f_j|e_1^{I_n})} C_n(e,e') \tag{5}$$

and

$$C_n(e,e') = \sum_{i=1}^{I_n} \sum_{k=i+1}^{I_n} \delta(e,e_i)\delta(e',e_k).$$

The function $\delta(\cdot,\cdot)$ denotes the Kronecker delta. The resulting training procedure is analogous to the one presented in (Brown et al., 1993) and (Tillmann and Ney, 1997).

The next section presents variants of the basic unconstrained model by putting restrictions on the valid regions of triggers (in-phrase vs. out-of-phrase) and using alignments obtained from either GIZA++ training or forced alignments in order to reduce the model size and to incorporate knowledge already obtained in previous training steps.

3.2 Model variations

Based on the unconstrained triplet model presented in Section 3, we introduce additional constraints, namely the *phrase-bounded* and the *path-aligned* triplet model in the following. The former reduces the number of possible triplets by posing constraints on the position of where valid triggers may originate from. In order to obtain phrase boundaries on the training data, we use forced alignments, i.e. translate the whole training data by constraining the translation hypotheses to the target sentences of the training corpus.

Path-aligned triplets use an alignment constraint from the word alignments that are trained with GIZA++. Here, we restrict the first trigger pair (f, e) to the alignment path as based on the alignment matrix produced by IBM model 4.

These variants require information in addition to the bilingual sentence pair (f_1^J, e_1^I), namely a corresponding phrase segmentation $\Pi = \{\pi_{ij}\}$ with

$$\pi_{ij} = \begin{cases} 1 & \exists \text{ a phrase pair that covers } e_i \text{ and } f_j \\ 0 & \text{otherwise} \end{cases}$$

for the phrase-bounded method and, similarly, a word alignment $A = \{a_{ij}\}$ where

$$a_{ij} = \begin{cases} 1 & \text{if } e_i \text{ is aligned to } f_j \\ 0 & \text{otherwise} \end{cases} .$$

3.2.1 Phrase-bounded triplets

The phrase-bounded triplet model (referred to as p_{phr} in the following), restricts the first trigger e to the same phrase as f, whereas the second trigger e' is set outside the phrase, resulting in

$$p_{phr}(f_j|e_1^I,\Pi) = \frac{1}{Z_j} \sum_{i=1}^{I} \sum_{k=1}^{I} \pi_{ij}(1-\pi_{kj})\alpha(f_j|e_i,e_k). \tag{6}$$

3.2.2 Path-aligned triplet

The path-aligned triplet model (denoted by p_{align} in the following), restricts the scope of e to words aligned to f by A, yielding:

$$p_{align}(f_j|e_1^I,A) = \frac{1}{Z_j} \sum_{i=1}^{I} \sum_{k=1}^{I} a_{ij}\alpha(f_j|e_i,e_k) \tag{7}$$

where the Z_j are, again, the appropriate normalization terms.

Also, to account for non-aligned words (analogously to the IBM models), the empty word e_0 is considered in all three model variations. We show the effect of the empty word in the experiments (Section 4). Furthermore, we can train the presented models in the inverse direction, i.e. $p(e|f,f')$, and combine the two directions in the rescoring framework. The next section presents a set of experiments that evaluate the performance of the presented triplet model and its variations.

4 Experiments

In this section, we describe the system setup used in this work, including the translation tasks and the corresponding training corpora. The experiments are based on an n-best list reranking framework.

4.1 System

The experiments were carried out using a state-of-the-art phrase-based SMT system. The dynamic programming beam search decoder uses several models during decoding by combining them log-linearly. We incorporate phrase translation and word lexicon models in both directions, a language model, as well as phrase and word penalties including a distortion model for the reordering. While generating the hypotheses, a word graph is created which compactly represents the most likely translation hypotheses. Out of this word graph, we generate n-best lists and use them to test the different setups as described in Section 3.

In the experiments, we use 10,000-best lists containing unique translation hypotheses, i.e. duplicates generated due to different phrase segmentations are reduced to one single entry. The advantage of this reranking approach is that we can directly test the obtained models since we already have fully generated translations. Thus, we can apply the triplet lexicon model based on $p(f|e, e')$ and its inverse counterpart $p(e|f, f')$ directly. During decoding, since e' could be from anywhere outside the current phrase, i.e. even from a part which lies beyond the current context which has not yet been generated, we would have to apply additional constraints during training (i.e. make further restrictions such as $i' < i$ for a trigger pair $(e_i, e_{i'})$).

Optimization of the model scaling factors is carried out using minimum error rate training (MERT) on the development sets. The optimization criterion is 100-BLEU since we want to maximize the BLEU score.

4.2 Tasks

4.2.1 IWSLT

For the first part of the experiments, we use the corpora that were released for the IWSLT'07 evaluation campaign. The training corpus consists of approximately 43K Chinese-English sentence pairs, mainly coming from the BTEC corpus (Basic Travel Expression Corpus). This is a multilingual speech corpus which contains tourism-related material, such as transcribed conversations about making reservations, asking for directions or conversations as taking place in restaurants. For the experiments, we use the clean data track, i.e. transcriptions of read speech. As the development set which is used for tuning the parameters of the baseline system and the reranking framework, we use the IWSLT'04 evaluation set (500 sentence pairs). The two blind test sets which are used to evaluate the final performance of the models are the official evaluation sets from IWSLT'05 (506 sentences) and IWSLT'07 (489 sentences).

The average sentence length of the training corpus is 10 words. Thus, the task is somewhat limited and very domain-specific. One of the advantages of this setting is that preliminary experiments can be carried out quickly in order to analyze the effects of the different models in detail. This and the small vocabulary size (12K entries) makes the corpus ideal for first "rapid application development"-style setups without having to care about possible constraints due to memory requirements or CPU time restrictions.

4.2.2 EPPS

Furthermore, additional experiments are based on the EPPS corpus (European Parliament Plenary Sessions) as used within the FTE (Final Text Edition) track of the TC-STAR evaluations. The corpus contains speeches held by politicians at plenary sessions of the European Parliament that have been transcribed, "corrected" to make up valid written texts and translated into several target languages. The language pairs considered in the experiments here are Spanish-English and English-Spanish.

The training corpus consists of roughly 1.3M sentence pairs with 35.5M running words on the English side. The vocabulary sizes are considerably larger than for the IWSLT task, namely around 170K on the target side. As development set, we use the development data issued for the 2006 evaluation (1122 sentences), whereas the two blind test sets are the official evaluation data from 2006 (TC-Star'06, 1117 sentences) and 2007 (TC-Star'07, 1130 sentences).

4.3 Results

4.3.1 IWSLT experiments

One of the first questions that arises is how many EM iterations should be carried out during training of the triplet model. Since the IWSLT task is small,

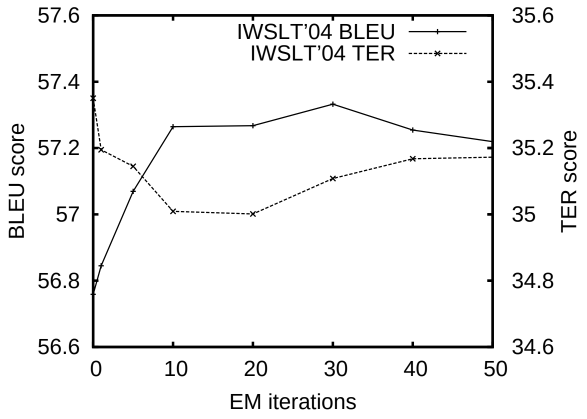


Figure 2: Effect of EM iterations on IWSLT'04, left axis shows BLEU (higher numbers better), right axis (dashed graph) shows TER score (lower numbers better).

	IWSLT'04		IWSLT'05	
	BLEU	TER	BLEU	TER
baseline	56.7	35.49	61.1	30.59
$p_{all}(e\|f,f')$	57.1	35.03	61.3	30.55
w/ singletons	57.3	35.04	61.3	30.61
w/ empties	57.3	35.00	61.2	30.65
+ $p_{all}(f\|e,e')$	**57.5**	**34.69**	**61.7**	**30.24**

Table 1: Different setups showing the effect of singletons and empty words for IWSLT CE IWSLT'04 (dev) and IWSLT'05 (test) sets, p_{all} triplets, 20 EM iterations.

	IWSLT'05		IWSLT'07	
	BLEU	TER	BLEU	TER
baseline	61.1	30.59	38.9	45.60
IBM model 1	61.5	30.29	39.4	45.31
trip fe+ef p_{all}	**61.7**	**30.24**	**39.7**	45.24
trip fe+ef p_{phr}	61.5	30.32	39.1	45.36
trip fe+ef p_{align}	61.2	30.60	**39.7**	**45.02**

Table 2: Comparison of triplet variants on IWSLT CE test sets, 20 EM iterations, with singletons and empty words.

we can quickly run the experiments on a full unconstrained triplet model without any cutoff or further constraints. Figure 2 shows the rescoring performance for different numbers of EM iterations. The first 10 iterations significantly improve the triplet model performance for the IWSLT task. After that, there are no big changes. The performance even degrades a little bit after 30 iterations. For the IWSLT task, we therefore set a fixed number of 20 EM iterations for the following experiments since it shows a good performance in terms of both BLEU and TER score. The oracle TER scores of the 10k-best lists are 14.18% for IWSLT'04, 11.36% for IWSLT'05 and 18.85% for IWSLT'07, respectively.

The next chain of experiments on the IWSLT task investigates the impact of changes to the setup of training an unconstrained triplet model, such as the addition of the empty word and the inclusion of singletons (i.e. triplets that were only seen once in the training data). This might show the importance of rare events in order to derive strategies when moving to larger tasks where it is not feasible to train all possible triplets, such as e.g. on the EPPS task (as shown later) or the Chinese-English NIST task. The results for the unconstrained model are shown in Table 1, beginning with a full triplet model in reverse direction, $p_{all}(e|f, f')$, that contains no singletons and no empty words for the triggering side. In this setting, singletons seem to help on dev but there is no clear improvement on one of the test sets, whereas empty words do not make a significant difference but can be used since they do not harm either. The baseline can be improved by +0.6% BLEU and around -0.5% in TER on the IWSLT'04 set. For the various setups, there are no big differences in the TER score which might be an effect of optimization on BLEU. Therefore, for further experiments using the constraints from Section 3.2, we use both singletons and empty words as the default.

Adding the other direction $p(f|e, e')$ results in another increase, with a total of +0.8% BLEU and -0.8% TER, which shows that the combination of both directions helps overall translation quality. The results on the two test sets are shown in Table 2. As can be seen, we arrive at similar improvements, namely +0.6% BLEU and -0.3% TER on IWSLT'05 and +0.8% BLEU and -0.4% TER on IWSLT'07, respectively. The constrained models, i.e. the phrase-bounded (p_{phr}) and path-aligned (p_{align}) triplets are outperformed by the full unconstrained case, although on IWSLT'07 both unconstrained and path-aligned models are close.

For a fair comparison, we added a classical IBM model 1 in the rescoring framework. It can be seen that the presented triplet models slightly outperform

	TC-Star'06		TC-Star'07	
	BLEU	TER	BLEU	TER
baseline	52.3	34.57	50.4	36.46
trip fe+ef p_{all}	52.9	34.32	50.6	36.34
+ max dist 10	52.9	34.20	50.8	36.22

Table 3: Effect of using maximum distance constraint for p_{all} on EPPS Spanish-English test sets, occ_3, 4 EM iterations due to time constraints.

the simple IBM model 1. Note that IBM model 1 is a special case of the triplet lexicon model if the second trigger is the empty word.

4.3.2 EPPS experiments

Since EPPS is a considerably harder task (larger vocabulary and longer sentences), the training of a full unconstrained triplet model cannot be done due to memory restrictions. One possibility to reduce the number of extracted triplets is to apply a maximum distance constraint in the training procedure, i.e. only trigger pairs are considered where the distance between first and second trigger is below or equal to the specified maximum.

Table 3 shows the effect of a maximum distance constraint for the Spanish-English direction. Due to the large amount of triplets (we extract roughly two billion triplets[2] for the EPPS data), we drop all triplets that occur less than 3 times which results in 640 million triplets. Also, due to time restrictions[3], we only train 4 iterations and compare it to 4 iterations of the same setting with the maximum distance set to 10. The training with the maximum distance constraints ends with a total of 380 million triplets. As can be seen (Table 3), the performance is comparable while cutting down the computation time from 9.2 to 3.1 hours. The experiments were carried out on a 2.2GHz Opteron machine with 16 GB of memory. The overall gain is +0.4–0.6% BLEU and up to -0.4% in TER. We even observe a slight increase in BLEU for the TC-Star'07 set which might be a random effect due to optimization on the development set where the behavior is the same as for TC-Star'06.

[2]Extraction can be easily done in parallel by splitting the corpus and merging identical triplets iteratively in a separate step for two chunks at a time.

[3]One iteration needs more than 12 hours for the unconstrained case.

	TC-Star'06		TC-Star'07	
	BLEU	TER	BLEU	TER
baseline	49.5	37.65	51.0	36.03
trip fe+ef p_{phr}	50.2	37.01	51.5	35.38
+ occ_2	50.2	37.06	51.8	35.32

Table 4: Results on EPPS, English-Spanish, p_{phr} combined, occ_3, 10 EM iterations.

	TC-Star'06		TC-Star'07	
	BLEU	TER	BLEU	TER
baseline	49.5	37.65	51.0	36.03
using FA	50.0	37.18	51.7	35.52
using IBM4	50.0	37.12	51.7	35.43
+ occ_2	50.2	36.84	52.0	35.10
+ max dist 1	50.0	37.10	51.7	35.51

Table 5: Results on EPPS, English-Spanish, maximum approximation, p_{align} combined, occ_3, 10 EM iterations.

Results on EPPS English-Spanish for the phrase-bounded triplet model are presented in Table 4. Since the number of triplets is less than for the unconstrained model, we can lower the cutoff from 3 to 2 (denoted in the table by occ_3 and occ_2, respectively). There is a small additional gain on the TC-Star'07 test set by this step, with a total of +0.7% BLEU for TC-Star'06 and +0.8% BLEU for TC-Star'07.

Table 5 shows results for a variation of the path-aligned triplet model p_{align} that restricts the first trigger to the best aligned word as estimated in the IBM model 1, thus using a maximum-approximation of the given word alignment. The model was trained on two word alignments, firstly the one contained in the forced alignments on the training data, and secondly on an IBM-4 word alignment generated using GIZA++. For this second model we also demonstrate the improvement obtained when increasing the triplet lexicon size by using less trimming.

Another experiment was carried out to investigate the effect of immediate neighboring words used as triggers within the p_{align} setting. This is equivalent to using a "maximum distance of 1" constraint. We obtained worse results, namely a 0.2-0.3% drop in BLEU and a 0.3-0.4% raise in TER (cf. Table 5, last row), although the training is significantly faster with this setup, namely roughly 30 minutes per it-

	TC-Star'06		TC-Star'07	
	BLEU	TER	BLEU	TER
baseline	49.5	37.65	51.0	36.03
IBM model 1	50.0	37.12	51.8	35.51
p_{all}, occ_3	50.0	37.17	51.8	35.43
p_{phr}, occ_2	50.2	37.06	51.8	35.32
p_{align}, occ_2	**50.2**	**36.84**	**52.0**	**35.10**

Table 6: Final results on EPPS English-Spanish, constrained triplet models, 10 EM iterations, compared to standard IBM model 1.

f	e	e'	$\alpha(f\|e,e')$
pagar	taxpayer	bill	0.76
factura	taxpayer	bill	0.11
contribuyente	taxpayer	bill	0.10
f	e	–	$p_{ibm1}(f\|e)$
contribuyente	taxpayer		0.40
contribuyentes	taxpayer		0.18
europeo	taxpayer		0.08
factura	bill		0.19
ley	bill		0.18
proyecto	bill		0.11

Table 7: Example of triplets and related IBM model 1 lexical probabilities. The triggers "taxpayer" and "bill" have a new effect ("pagar"), previously not seen in the top ranks of the lexicon.

eration using less than 2 GB of memory. However, this shows that triggers outside the immediate context help overall translation quality. Additionally, it supports the claim that the presented methods are a complementary alternative to the WSD approaches mentioned in Section 2 which only consider the immediate context of a single word.

Finally, we compare the constrained models to an unconstrained setting and, again, to a standard IBM model 1. Table 6 shows that the p_{align} model constrained on using the IBM-4 word alignments yields +0.7% in BLEU on TC-Star'06 which is +0.2% more than with a standard IBM model 1. TER decreases by -0.3% when compared to model 1. For the TC-Star'07 set, the observations are similar.

The oracle TER scores of the development n-best list are 25.16% for English-Spanish and 27.0% for Spanish-English, respectively.

5 Discussion

From the results of our reranking experiments, we can conclude that the presented triplet lexicon model outperforms the baseline single-best hypotheses of the decoder. When comparing to a standard IBM model 1, the improvements are significantly smaller though measurable. So far, since IBM model 1 is considered one of the stronger rescoring models, these results look promising. An unconstrained triplet model has the best performance if training is feasible since it also needs the most memory and time to be trained, at least for larger tasks.

In order to cut down computational requirements, we can apply phrase-bounded and path-aligned training constraints that restrict the possibilities of selecting triplet candidates (in addition to simple thresholding). Although no clear effect could be observed for adding empty words on the triggering side, it does not harm and, thus, we get a similar functionality to IBM model 1 being "integrated" in the triplet lexicon model. The phrase-bounded training variant uses forced alignments computed on the whole training data (i.e. search constrained to producing the target sentences of the bilingual corpus) but could not outperform the path-aligned model which reuses the alignment path information obtained in regular GIZA++ training.

Additionally, we observe a positive impact from triggers lying outside the immediate context of one predecessor or successor word.

5.1 Examples

Table 7 shows an excerpt of the top entries for $(e, e') = (taxpayer, bill)$ and compares it to the top entries of a lexicon based on IBM model 1. We observe a triggering effect since the Spanish word *pagar* (*to pay*) is triggered at top position by the two English words *taxpayer* and *bill*. The average distance of *taxpayer* and *bill* is 5.4 words. The models presented in this work try to capture this property and apply it in the scoring of hypotheses in order to allow for better lexical choice in specific contexts.

In Table 8, we show an example translation where rescoring with the triplet model achieves higher n-gram coverage on the reference translation than the variant based on IBM model 1 rescoring. The differing phrases are highlighted.

Source sentence	…respecto de la Posición Común del Consejo con vistas a la adopción del Reglamento del Parlamento Europeo y del Consejo relativo al …
IBM-1 rescoring	…on the Council common position with a view to the adoption of the Rules of Procedure of the European Parliament and of the Council …
Triplet rescoring	…on the common position of the Council with a view to the adoption of the regulation of the European Parliament and of the Council …
Reference translation	…as regards the Common Position of the Council with a view to the adoption of a European Parliament and Council Regulation as regards the …

Table 8: A translation example on TC-Star'07 Spanish-English comparing the effect of the triplet model to a standard IBM-1 model.

6 Outlook

We have presented a new lexicon model based on triplets extracted on a sentence level and trained iteratively using the EM algorithm. The motivation of this approach is to add an additional second trigger to a translation lexicon component which can come from a more global context (on a sentence level) and allow for a more fine-grained lexical choice given a specific context. Thus, the method is related to word sense disambiguation approaches.

We showed improvements by rescoring n-best lists of the IWSLT Chinese-English and EPPS Spanish-English/English-Spanish task. In total, we achieve up to +1% BLEU for some of the test sets in comparison to the decoder baseline and up to +0.3% BLEU compared to IBM model 1.

Future work will address an integration into the decoder since the performance of the current rescoring framework is limited by the quality of the n-best lists. For the inverse model, $p(e|f, f')$, an integration into the search is directly possible. Further experiments will be conducted, especially on large tasks such as the NIST Chinese-English and Arabic-English task. Training on these huge databases will only be possible with an appropriate selection of promising triplets.

Acknowledgments

This material is partly based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023, and was partly realized as part of the Quaero Programme, funded by OSEO, French State agency for innovation.

The authors would like to thank the anonymous reviewers for their valuable comments.

References

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

Marine Carpuat and Dekai Wu. 2007. Improving statistical machine translation using word sense disambiguation. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, Prague, Czech Republic, June.

Yee Seng Chan, Hwee Tou Ng, and David Chiang. 2007. Word sense disambiguation improves statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 33–40, Prague, Czech Republic, June. Association for Computational Linguistics.

Ciprian Chelba and Frederick Jelinek. 2000. Structured language modeling. *Computer Speech and Language*, 14(4):283–332.

Stephen A. Della Pietra, Vincent J. Della Pietra, John R. Gillett, John D. Lafferty, Harry Printz, and Luboš Ureš. 1994. Inference and estimation of a long-range trigram model. In J. Oncina and R. C. Carrasco, editors, *Grammatical Inference and Applications, Second International Colloquium, ICGI-94*, volume 862, pages 78–92, Alicante, Spain. Springer Verlag.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–22.

Woosung Kim and Sanjeev Khudanpur. 2003. Cross-lingual lexical triggers in statistical language modeling. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 17–24, Morristown, NJ, USA. Association for Computational Linguistics.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

Ronald Rosenfeld. 1996. A maximum entropy approach to adaptive statistical language modeling. *Computer Speech and Language*, 10(3):187–228.

Christoph Tillmann and Hermann Ney. 1997. Word triggers and the EM algorithm. In *Proc. Special Interest Group Workshop on Computational Natural Language Learning (ACL)*, pages 117–124, Madrid, Spain, July.

Christoph Tillmann. 2001. *Word Re-Ordering and Dynamic Programming based Search Algorithm for Statistical Machine Translation*. Ph.D. thesis, RWTH Aachen University, Aachen, Germany, May.

A Casual Conversation System Using Modality and Word Associations Retrieved from the Web

Shinsuke Higuchi **Rafal Rzepka** **Kenji Araki**
Graduate School of Information Science and Technology
Hokkaido University , Sapporo Japan 060-0814
{shin_h,kabura,araki}@media.eng.hokudai.ac.jp

Abstract

In this paper we present a textual dialogue system that uses word associations retrieved from the Web to create propositions. We also show experiment results for the role of modality generation. The proposed system automatically extracts sets of words related to a conversation topic set freely by a user. After the extraction process, it generates an utterance, adds a modality and verifies the semantic reliability of the proposed sentence. We evaluate word associations extracted form the Web, and the results of adding modality. Over 80% of the extracted word associations were evaluated as correct. Adding modality improved the system significantly for all evaluation criteria. We also show how our system can be used as a simple and expandable platform for almost any kind of experiment with human-computer textual conversation in Japanese. Two examples with affect analysis and humor generation are given.

1 Introduction

Many task-oriented dialogue systems (Liu et al., 2003; Reitter et al., 2006) have been developped. Research on non-task-oriented dialogue systems like casual conversation dialogue systems (”chatbots”) is on the other hand not very common, perhaps due to the many amateurs who try to build naturally talking systems using sometimes very clever, but rather unscientific methods although there are systems with chatting abilities as (Bickmore and Cassell, 2001) but concentrate on applying strategies to casual conversation rather than their automatic generation of those conversations. However, we believe that the main reason is that an unrestricted domain is disproportionately difficult compared to the possible use such a system could have. It is for example very hard to predict the contents and topics of user utterances, and therefore it is almost impossible to prepare conversational scenarios. Furthermore, scenarios need more or less specific goals to be useful. However in our opinion, sooner or later non-task-oriented dialogue systems will have to be combined with task oriented systems and used after recognizing that the user's utterance does not belong to a given task. This would lead to more natural interfaces for e.g. information kiosks or automatic guides placed in public places where anyone can talk to them about anything (Gustafson and Bell, 2000; Kopp et al., 2005) regardless of the role the developers intended. For this reason we have also started implementing emotiveness recognition and joke generation modules that are presented later in the paper.

Well-known examples of non-task-oriented dialogue systems are ELIZA (Weizenbaum, 1966) and A.L.I.C.E [1], though the former was built to parody a Rogerian therapist which can be regarded as a task. Both systems and their countless imitators [2] use a lot of rules coded by hand. ELIZA is able to make a response to any input, but these responses are only information requests without providing any new information to the user. In the case of A.L.I.C.E,

[1]Wallace, R. The Anatomy of A.L.I.C.E. http://www.alicebot.org/anatomy.html.

[2]Many of them have been quite successful in the Loebner Prize and the Chatterbox Challenge (competitions only for English-speaking bots) but explanations of their algorithms are not available.

the knowledge resource is limited to the existing database. Creating such databases is costly and a programmer must learn the AIML mark-up language to build it. Although there have been attempts at updating AIML databases automatically (Pietro et al., 2005), the scale was rather limited.

As mentioned above, these examples and many other ”chatbots” need hand-crafted rules, and are thus often ignored by computer scientists and rarely become a research topic. However, they have proved to be useful for e-learning (Pietro et al., 2005) and machine learning (Araki and Kuroda, 2006) support.

Building a system using automatic methods, like we do, seems to be the most realistic way for unrestricted domains. Considering the large cost of developing a program that can talk about any topic, it is appealing to turn to the huge and cheap textual source that is the Internet.

In this very moment millions of people (Kumar et al, 2003) are updating their blogs and writing articles on every possible topic. These are available on the Web which we can access any time, and in a faster and faster manner, the search engines grow more and more efficient. Thus, the Web is well suited to extracting word associations triggered by words from user utterances made in a topic-free dialogue system.. We present a system making use of this type of information. It automatically extracts word association lists using all keywords in a given utterance without choosing a specific one (which most other systems that ignore the context do) then generates a reply using the only one strongest association from the nouns, verbs and adjectives association groups. Modality is then added to the reply, and then it is output.

Our system is built upon the idea that human utterances consist of a proposition and a modality (Nitta et al., 1989). In this paper we present an algorithm for extracting word associations from the Web and a method for adding modality to statements. We evaluate both the word associations and the use of modality. We also suggest some future possible extensions of the system and show a small experiment with adding humor to the system.

In this paper, the system described works for Japanese and uses text as input and output. Though the final goal of our research is to help developing freely talking car navigation systems that by their chatting abilities can help to avoid drowsiness while driving and so on. in this part of the development we concentrate on proposition generation and modality processing. Therefore, we work only with text now. We plan to combine this project with research on in car voice recognition and generation.

2 Extracting Word Associations

In this chapter, we present a method for automatic extraction of word associations based on keywords from user utterances. We use the Google[3] search engine snippets to extract word associations in real time without using earlier prepared resources, such as off-line databases.

2.1 Extracting Word Associations from the Web

In the first step, the system analyzed user utterances using the morphological analyzer MeCab[4] in order to spot query keywords for extracting word associations lists. We define nouns, verbs, adjectives, and unknown words as query keywords. The reason we chose these word classes is that these word classes can be treated as important and, to some extent, describe the context. We define a noun as the longest set of nouns in a compound noun. For example, the compound noun *shizen gengo shori*[5] (natural language processing) is treated by MeCab as three words: (*shizen* - natural), (*gengo* - language) and (*shori* - processing). Our system, however, threats it as one noun.

In the next step, the system uses these keywords as query words for the Google search engine. The system extracts the nouns from the search results and sorts them in frequency order. This process is based on the idea that words which co-occur frequently with the input words are of high relevance to them. The number of extracted snippets is 500. This value was set experimentally, taking the processing time and output quality into account. The top ten words of a list are treated as word associations, see Table 1 for an example.

[3]Google, http://www.google.co.jp/

[4]MeCab: Yet Another Part-of-Speech and Morphological Analyzer, http://mecab.sourceforge.jp/

[5]All Japanese transcriptions will be written in italics.

Table 1: Examples of noun associations triggered by a user utterance

Sapporo wa samui. (Sapporo city is cold.)		
Association frequency ranking:		
1	*yuki* (snow)	52
2	*fuyu* (winter)	50
3	*kion* (temperature)	16
4	*jiki* (season)	12
5	*Tokyo* (Tokyo)	12
6	*tenki* (weather)	11
7	*chiiki* (area)	10
8	*heya* (room)	10

2.2 Evaluation

We asked volunteers to use our system and to evaluate the correctness of word lists generated by the system. First, a participant freely inputs an utterance, for which the system retrieves ten association words. Next, a participant rated these words using a scale of one to three with 3 meaning ”perfectly correct”, 2 -”partially correct” and 1 - ”incorrect”. In this experiment we consider words that receive a 2 or 3 as usable. The reason associations rated 2 or 3 are considered as usable is that the definition of what makes a good word association here is difficult to specify. When it comes to topic-free conversations we have observed that associations have an effect on a certain context. Three volunteers repeated the experiment ten times, so the final amount of evaluated words was 300. Table 2 shows the results of the top 10 words, sorted by the frequency of appearance. Table 3 shows the results of the top 5 words.

What constitutes a correct word association was left to each volunteer to decide subjectively since in a casual conversation setting associations are hard to define strictly.

Table 2: Top 10 word associations

score	participant(A，B，C)	total
3	40，52，57	149
2	37，17，27	81
1	23，31，16	70
usability (%)	77，69，84	77

As shown in Table 2 approximately 77% of the word associations were judged as usable but there were individual differences between the evaluators. This shows that the definition of word associations is different for each participant. Table 3 shows that approximately 80% of the word associations were judged as usable. It is thus highly likely that the top words from the frequency lists are correct associations. The results show that automatic extracting of word associations using a Web search engine is feasible. The main reason for extracting word associations from the Web is that thanks to this method, the system can handle new information, proper names, technical terms and so on. by using only the snippets from the search engine. The word association extraction takes no more than few seconds. For the evaluation we used only nouns but we expect although verbs and adjectives are often more abstract than nouns, the word associations for them will improve the results.

Table 3: Top 5 word associations

score	participant（A，B，C）	total
3	20，29，36	85
2	17，9，10	36
1	13，12，4	29
usability (%)	74，76，92	81

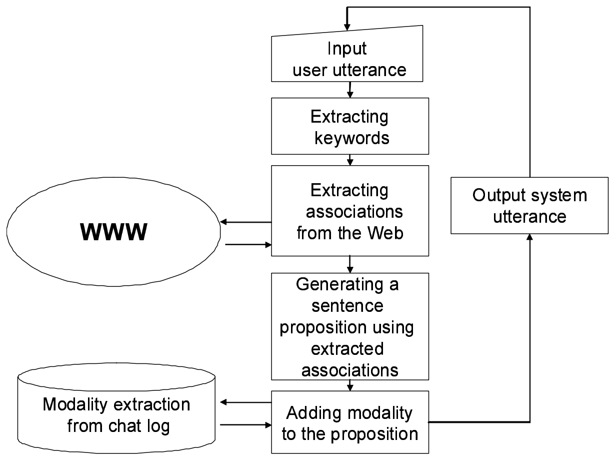
3 General Description of the System

The system generates replies in the following way:

- extraction of keywords from user utterance
- extraction of word associations from the Web
- generation of sentence proposition using the extracted associations
- addition of modality to the sentence proposition

3.1 Extraction of Keywords from User Utterances

The system applies morphological analysis to the use utterances in the same way as described in section **2.1** and extracts keywords based on part of speech.



Input user utterance
Extracting keywords
Extracting associations from the Web
WWW
Output system utterance
Generating a sentence proposition using extracted associations
Modality extraction from chat log
Adding modality to the proposition

Figure 1: System flow

3.2 Extraction of Words Association from the Web

The system performs a Google search using the extracted keywords as a query. The system sorts the results obtained from the query by their frequency as in section **2.1**. In section **2.1** only nouns were extracted but here we also extract verbs and adjectives. After sorting all words in adjective, verb and noun lists the system uses the ones with the highest frequency as word associations.

3.3 Generation of Proposition Using Word Associations

Using the associations, the system generates the proposition of a sentence to be used as a reply to the user input. A proposition is an expression representing an objective statement. The proposition is generated by applying associations to a proposition template like [(noun) (topic indicating particle *wa*) (adjective)]. We prepared 8 proposition templates manually (see Table 4). The templates were chosen subjectively after examining statistics from IRC [6] chat logs. Our criteria for choosing templates from the chat logs was that they should belong to the 20 most frequent modality patterns and to be flexible enough to fit a range of grammatical constructions, for example in English, ”isn’t it” cannot follow verbs while ”I guess” can follow nouns, adjectives, and verbs. The proposition templates are applied in a predetermined order: for example, first a template ”(noun) *(wa)* (adjective)” is used; next a template ”(noun) *(ga)* (adjective)” is used. However, since the generated proposition is not always a natural statement, the system uses exact matching searches of the whole phrases in a search engine to check the naturalness of each proposition. If the frequency of occurrence of the proposition is low, it is defined as unnatural and deleted. This processing is based on the idea that the phrases existing on the Web in large numbers are most probably correct grammatically and semantically. If an unnatural proposition is generated, the system generates another proposition in the same way. In this experiment the system used propositions for which the hit number exceeded 1,000 hits using Google. Thus, the processing proceeds as follows. The system first selects the top noun, top verb, and top adjective word associations. These are applied to the templates in a predetermined order. If a generated proposition is judged as valid (using Google, occurrence on the web indicates validity), it is used. If not, another template is tried until a valid proposition is found. The reason for not trying every possible combination of associated words is prohibitively long processing time.

[6]Internet Relay Chat Protocol,
http://www.irchelp.org/irchelp/rfc/rfc.html

Table 4: Proposition templates

(noun) *(wa)* (adjective)
(noun) *(ga)* (adjective)
(noun) *(ga)* (verb)
(noun) *(wa)* (verb)
(so-re) *(wa)* (verb)
(noun)
(adjective)
(verb)

3.4 Adding Modality to the Propositions

Finally, the system adds modality to the generated proposition. By modality we mean a set of grammatical and pragmatic rules to express subjective judgments and attitudes. In our system, modality is realized through adverbs at the end of a sentence which is common in Japanese (Nitta et al., 1989). In our system, a pair of sentence head and sentence end auxiliary verb are defined as ”modality”.

3.4.1 Extracting Modality

There is no standard definition of what constitutes modality in Japanese. In this paper modality of casual conversation is classified into questions and informative expressions. Questions are expressions that request information from the user. Informative expressions are expressions that transmit information to the user. Patterns for these modalities are extracted automatically from IRC chat logs (100,000 utterances) in advance. Modality patterns are extracted in these ways:

- pairs of grammatical particles and an auxiliary verbs placed at the end of sentences are defined as ending patterns
- sentences with question marks are defined as questions
- adverbs, emotive words, and connectives at the beginning of sentences are defined as informative expressions
- candidate patterns thus obtained are sorted by frequency

First the system extracts sentence ending patterns from IRC chat logs. If an expression contains question marks, it is classified as a question. Next, the system extracts adverbs, emotive words, and connectives from the beginning and end of sentences from the IRC logs. These pairs (beginning and end) of expressions are classified as ”informative expressions”. For example question expression ”desu-ka?” is extracted from a human utterance like ”*Kyou-wa samui desu-ka?*” (Is it cold today?). An informative expression ”*maa *** kedo*” is extracted from a human utterance as ”*Maa sore-wa ureshii kedo*” (Well, I’m glad, but you know...).

685 patterns were obtained for informative expressions. 550 of these informative expression patterns were considered by authors as correct (80%). For questions 396 patterns were obtained, and 292 patterns (73%) were evaluated as correct. We sorted these candidates in frequency order. The words appearing at the top of the list were correct, but even the ones appearing only once were still deemed as usable. For example, the question expression ”*janakatta deshita-kke?*” is a correct expression, but appeared only once in the 100,000 utterances. Hence, we confirmed that chat logs include various modality expressions, and only a few of them are incorrect. Tables 5 and 6 show some examples of modality patterns.

Table 5: Examples of informative expression modality

informative expression	frequency
maa - kedo (Well , it can be said - but -)	21
maa - dana (Well , it can be said -)	16
maa - desu-ga (Well , it appears that -)	16
soko-de - desu-yo (Here , it is said that -)	15
maa - da-ga (Well , it can be said - but -)	14
maa - desu-yo (Well , it is that -)	12

Table 6: Examples of question modality sentence endings

question	freqency
...desuka? (Is it that ... ?)	232
...kana? (Maybe ... ?)	90
...da-kke? (Is it right that ... ?)	87
...masu-ka? (Is it that ... ?)	69
...nano? (Is it that ... ?)	68
...toka? (... , isn’t it ?)	55

3.4.2 Adding Modality

The system adds the modality from section **3.4.1** to the proposition from section **3.3** to generate the system output. This process is based on the idea that human utterance consists of proposition and modality. A modality pattern is selected randomly. For example, if the system generates the proposition ”*fuyu wa samui* (Winter is cold.)” and selects the modality ”*iyaa ... desu-yo* (Ooh ... isn’t it?)”, the generated output will be ”*iyaa, fuyu-wa samui desu-yo* (Winter is cold, you know)”. However, there is a possibility that the system generates unnatural output like ”*fuyu-wa samui dayo-ne* (Winter is cold, arent't it?)”, depending on the pair of proposition and modality. To this problem, the system uses the Google search engine to filter out unnatural output. The system performs a phrase search on the end of the sentence. If the number of search hits is higher than threshold, the output is judged as correct. If the number of a search hits is lower than the threshold, the output is judged as incorrect and discarded, and a new reply is generated. Here, we experimentally set the threshold to 100 hits.

4 Experimental Results

We used system α, generating only the proposition, and system β, generating both proposition and modality. 5 participants used each systems for conversations of 10 turns and evaluated the conversations on a 5-point scale. Evaluation criteria were ”will to continue the conversation” (A), ”grammatical naturalness of dialogues” (B), ”semantical naturalness of dialogues” (C), ”vocabulary richness” (D), ”knowledge richness” (E), and ”humanity of the system” (F). Table 7 shows average scores for the evaluations of each system. System β that uses modality scored much higher than system α. Table 8 shows examples of actual dialogue. In the evaluation, the participants expressed the opinion that an utterance like *(xx ha yy)* is unnatural and using a modality like *(maa)*(”well”), *(moo)*(”anyway”) is very natural. Thus we can say that the modality expressions make the utterances of the system seem more natural.

Table 8: Examples of dialogues with system β

user:	*Nanika suki-na tabemono aru?* (What food do you like ?)
system:	*Maa, tabemono-wa oishii desu.* (Well, food tastes good.)
user:	*Saikin-wa osake-mo suki-ni natte-kita-n-da.* (Recently, I begin to like alcohol, too.)
system:	*Demo, sake-wa yowai-no-yo-ne.* (But, I can't drink much.)
user:	*Yowai-n-da... Sugu yochau-no?* (Really...? Do you get drunk quickly?)
system:	*Maa, sore-wa sugoi-wa-ne.* (Whow, that's amazing.)
user:	*Boku-wa wari-to osake-wa tsuyoi hou-nan-da.* (I have a rather high tolerance for alcohol.)
system:	*Iya, sore-wa tsuyoi-n-de.* (Well, it is strong, I can say.)

user:	*Kyou-wa sukoshi hadazamui-yo-ne. Natsu-nanoni.* (It is a little cold today. Even though it is summer.)
system:	*Asa-wa hadazamui-toka ?* (Isn't it chilly in morning?)
user:	*Asa-kara hadazamukatta-ne.* (It was chilly since morning.)
system:	*Aa kyou-wa hayai-na.* (Ah, it's early.)

5 The System as a Platform

The simplicity, real-time processing capabilities and promising results showing that users do not get bored so quickly encouraged us to perform trials with other ongoing projects and experiment with the system working as a platform for adding various modules and algorithms. By using our system it is possible to perform tests to see if a new idea will support or improve human-computer interaction or not. Here we will briefly describe two such trials - one on guessing emotive values of utterances and one on improving the system's overall evaluation by adding a pun generator.

5.1 Testing Affect Analysis

Ptaszynski et al.(Ptaszynski et al., 2008) have developed a method for affect analysis of Japanese text. Their method is based on cross-referencing lexical emotive elements with emotive expressions appearing in text. In the process of analysis first a general emotive context is determined and then the specific types of emotional states conveyed in an utterance are extracted. They support this method with a

Table 7: Evaluation Results

	System α (proposition)						system β (proposition + modality)					
Evaluation criteria	A	B	C	D	E	F	A	B	C	D	E	F
Participant a	1	3	2	2	4	2	4	4	3	4	3	5
Participant b	1	3	1	2	1	1	4	4	4	5	4	3
Participant c	1	2	1	2	1	1	1	2	1	2	1	1
Participant d	1	3	1	3	1	2	4	3	1	3	3	4
Oarticipant e	1	4	1	1	2	1	3	2	2	4	5	4
Average	1.0	3.0	1.2	2.0	1.8	1.4	3.2	3.0	2.2	3.6	3.2	3.4

Web-mining technique to improve the performance of the emotional state type extraction. A system constructed on the basis of their method achieved human level performance in determining the emotiveness of utterances, and 65% of human level performance in extracting the specific types of emotions. Also, the supporting Web mining technique improved the performance of the emotional state type extraction to 85% of the human level (Shi et al, 2008). As these are very promising figures we are currently in the phase of implementing their ideas in our system and testing how emotion recognition can influence speech act analysis and the automatic choice of proper modality.

5.2 Improving the System Using Humor

In this trial, an experiment showing that humor can improve a non-task oriented conversational system's overall performance was conducted.

5.2.1 Implementing PUNDA system

By using a simplified version of Dybala's PUNDA system (Dybala et al., 2008), a pun-generation was added to our baseline system. The PUNDA algorithm consists of two parts: A Candidate Selection Algorithm and a Sentence Integration Engine. The former generates a candidate for a pun analyzing an input utterance and selects words or phrases that could be transformed into a pun by one of four generation patterns: homophony, initial mora addition, internal mora addition or final mora addition. The latter part generates a sentence including the candidate extracted in the previous step. To make the system's response more related to the user's input, each sentence that included a joke started with the pattern "[base phrase] *to ieba*" ("Speaking of [base phrase]"). The remaining part of the sentence was extracted from the Web and the candidate was used as a query word and the list of sentences including this word was retrieved. Then the shortest sentence with an exclamation mark is selected as most jokes convey some emotions. When the candidate list was empty, the system selected one random pun from a pun database.

5.2.2 Experiment results

In the first experiment, 5 participants were asked to perform a 10-turn dialogue with two systems. After using both systems (baseline and humor-equipped), users were asked to evaluate both systems's performances by answering the following questions: A) Do you want to continue the dialogue?; B) Was the system's utterances grammatically natural?; C) Was the system's utterances semantically natural?; D) Was the system's vocabulary rich?; E) Did you get an impression that the system possesses any knowledge?; F) Did you get an impression that the system was human-like?; G) Do you think the system tried to make the dialogue more funny and interesting? and H) Did you find the system's utterances interesting and funny? Answers were given on a 5-point scale and the results are shown in Table 9.

A third-person evaluation experiment was also performed and again the humor-equipped system scored higher than the non-humor one. The question asked in this evaluation was: "Which dialogue do you find most interesting and funny?". Evaluators could choose between 3 options: Dialogue 1 (Baseline system first 3 turns), Dialogue 2 (Humor-equipped system, first 3 turns with system's third response replaced by pun generator's output) and Dia-

Table 9: Results of humor experiments

Evaluation Criteria	A	B	C	D	E	F	G	H
Baseline System	3.0	2.2	2.4	2.4	2.0	2.8	2.2	2.8
With pun generator	3.2	3.0	2.8	2.8	2.2	3.0	3.4	3.6

logue 3 (the first 3 turns of the baseline system with joking ability). Dialogue 1 and Dialogue 2 have the same input. Among 25 evaluators, only 5 (20%) responded that Dialogue 1 was most interesting and funny. 10 chose Dialogue 2 and the other 10 chose Dialogue 3 (40% respectively). This means that each of humor equipped dialogues received evaluations two times higher than non-humor dialogue.

5.3 A Toolkit for Conversation-Related Experiments

Our system can be also disassembled into a set of flexible tools which help students to experiment with dialogue processing. By using simple web-mining techniques we described, this dialogue engine is capable of automatic retrieval of associations which can be used to produce a whole range of utterances - for example by using the bottom, not the top of the associations list, one can examine how interesting or provocative the dialogue becomes. As the system has a cgi interface, the experiments are easy and any new feature (for instance a speech act choice menu) can be easily added. Such toolkit gives students an opportunity to experiment on a given aspect of dialogue processing without the need of building a conversation system from the scratch. There is also no need of laborious knowledge input and, as such open-domain oriented system generates new ”on topic” utterances, experiment subjects do not get bored quickly, which is always a problem while collecting conversation logs of human-machine interaction. A programmer also can freely choose between thousands of IRC logs utterances and Internet resources for the statistical trials, grammar patterns retrieval, speech acts analysis.

6 Conclusion and Future Work

In this research we investigated if word associations extracted automatically from the Web are reasonable (semantically on topic) and if they can be successfully used in non-task-oriented dialogue systems. We also implemented such a system extraction module. It is able to automatically generate in real-time responses to user utterances by generating a proposition and adding modality retrieved from IRC chat logs. We conducted evaluation experiments on the overall influence of the modality usage and it improved the system. Therefore we showed that it is possible to construct a dialogue system that automatically generates understandable on-topic utterances without the need of creating vast amounts of rules and data beforehand. We also confirmed that our system can be used as a experimental platform which can be easily used by other researchers to test their algorithms with a more unpredictible (and less boring) ”chatbot”, an important factor for long tiring sessions of human-computer conversation. Currently there are several projects which use the system described here as a platform for experiments and we introduced two of them - on joke generation and affect analysis.

There is still a lot of work left to be done. It is necessary for a non-task-oriented dialogue system to obtain not only word associations, but also different kinds of knowledge - of user’s preferences or of dialogue itself - for example conversational strategies. At this moment the system generates utterances by applying word associations to the proposition templates and adding modality. We also need to more deeply consider semantics, speech acts and context to create a more advanced system. Finally, the system needs to recognize not only keywords, but also user’s modality. We assume that the affect recognition mentioned above will help us to achieve this goal in near future and this is our next step. By opening the system’s code and giving others the opportunity of adding their own modules and changes we hope to solve remaining problems. In this paper we focus on the impact of adding modality to a system. Comparing the system to Japanese versions of ELIZA (already available) and ALICE (not available in Japanese yet) is also one of our next steps.

Acknowledgments

This work was partially supported by the Research Grant from the Nissan Science Foundation.

References

Bei Liu, Limin Du, Shuiyuan Yu. 2003 The method of building expectation model in task-oriented dialogue systems and its realization algorithms. Proceedings of Natural Language Processing and Knowledge Engineering:174-179

David Reitter, Johanna D. Moore, and Frank Keller. 2006. Priming of syntactic rules in task-oriented dialogue and spontaneous conversation. In Proc. 28th Annual Conference of the Cognitive Science Society (CogSci), Vancouver, Canada.

Timothy Bickmore and Justine Cassell. 2001 Relational Agents: A Model and Implementation of Building User Trust. Proceedings of Human Factors Computing Systems (SIGCHI'01): 396–403.

Joakim Gustafson and Linda Bell. 2000. Speech technology on trial: Experiences from the August system. *In Natural Language Engineering*, 1(1):1-15.

Stefan Kopp, Lars Gesellensetter, Nicole C. Kramer, and Ipke Wachsmuth. 2005. *A Conversational Agent as Museum Guide– Design and Evaluation of a Real-World Application.* Intelligent Virtual Agents, LNAI 3661:329-343.

Joseph Weizenbaum. 1966. ELIZA - computer program for the study of natural language communication between man and machine. *Commun. ACM*, vol.9, no.1:36-45.

Orlando De Pietro, Maurizio De Rose, and Giovanni Frontera. 2005. Automatic Update of AIML Knowledge Base in E-Learning Environment. In Proceedings of Computers and Advanced Technology in Education. , Oranjestad, Aruba, August:29-31.

Kenji Araki and Michitomo Kuroda. 2006. Generality of a Spoken Dialogue System Using SeGA-IL for Different Languages, Proceedings of the IASTED International Conference COMPUTER INTELLIGENCE:70-75.

Ravi Kumar, Jasmine Novak, Prabhakar Raghavan, and Andrew Tomkins. 2003. On the Bursty Evolution of Blogspace. Proceedings of The Twelfth International World Wide Web Conference:568-257

Yoshio Nitta and Takashi Masuoka, *Japanese modality(Nihongo no modality)* Kuroshio.

Michal Ptaszynski, Pawel Dybala, Rafal Rzepka, and Kenji Araki. 2008. Double Standpoint Evaluation Method for Affect Analysis System. The 22nd Annual Conference of Japanese Society for Artificial Intelligence (JSAI 2008).

Wenhan Shi, Rafal Rzepka and Kenji Araki. 2008. Emotive Information Discovery from User Textual Input Using Causal Associations from the Internet (in Japanese)", Proceedings of the 7th Forum of Information Technology(Vol2):267-268

Pawel Dybala, Michal Ptaszynski, Rafal Rzepka and Kenji Araki. 2008. Extracting Dajare Candidates from the Web - Japanese Puns Generating System as a Part of Humor Processing Research. Proceedings of LIBM'08 First International Workshop on Laughter in Interaction and Body Movement:46-51.

When Harry Met Harri, هاري and 亨利: Cross-lingual Name Spelling Normalization

Fei Huang , Ahmad Emami and Imed Zitouni

IBM T. J. Watson Research Center

1101 Kitchawan Road
Yorktown Heights, NY 10598
`{huangfe, emami, izitouni}@us.ibm.com`

Abstract

Foreign name translations typically include multiple spelling variants. These variants cause data sparseness problems, increase Out-of-Vocabulary (OOV) rate, and present challenges for machine translation, information extraction and other NLP tasks. This paper aims to identify name spelling variants in the target language using the source name as an anchor. Based on word-to-word translation and transliteration probabilities, as well as the string edit distance metric, target name translations with similar spellings are clustered. With this approach tens of thousands of high precision name translation spelling variants are extracted from sentence-aligned bilingual corpora. When these name spelling variants are applied to Machine Translation and Information Extraction tasks, improvements over strong baseline systems are observed in both cases.

1 Introduction

Foreign names typically have multiple spelling variants after translation, as seen in the following examples:

```
He confirmed that "al-Kharroub
province is at the top of our
priorities."
…for the Socialist Progressive
Party in upper Shuf and the Al-
Kharrub region,…
…during his tour of a number of
villages in the region of Al-
Kharub,…
…Beirut and its suburbs and
Iqlim al-Khurub,…
```

Such name spelling variants also frequently appear in other languages, such as 布什(bushi) / 布殊(bushu) / 布希(buxi) (for Bush) in Chinese, and سبرنغفيلد (sbrngfyld) /سبرينغفيلد (sbryngfyld) / سبرينجفيلد (sbrynjfyld) (for Springfield) in Arabic.

These spelling variants present challenges for many NLP tasks, increasing vocabulary size and OOV rate, exacerbating the data sparseness problem and reducing the readability of MT output when different spelling variants are generated for the same name in one document. We address this problem by replacing each spelling variant with its corresponding canonical form. Such text normalization could potentially benefit many NLP tasks including information retrieval, information extraction, question answering, speech recognition and machine translation.

Research on name spelling variants has been studied mostly in Information Retrieval research, especially in query expansion and cross-lingual IR. Baghat and Hovy (2007) proposed two approaches for spelling variants generation, based on the letters-to-phonemes mapping and Soundex algorithm (Knuth 1973). Raghaven and Allan (2005) proposed several techniques to group names in ASR output and evaluated their effectiveness in spoken document retrieval (SDR). Both approaches use a named entity extraction system to automatically identify names. For multi-lingual name spelling variants, Linden (2005) proposed to use a general edit distance metric with a weighted FST to find technical term translations (which were referred to as "cross-lingual spelling variants"). These variants are typically translated words with similar stems in another language. Toivonen and colleagues (2005) proposed a two-step fuzzy translation technique to solve similar problems. Al-Onaizan and Knight (2002), Huang (2003) and Ji and Grishman (2007) investigated the general name entity translation problem, especially in the context of machine translation.

This paper aims to identify mono-lingual name spelling variants using cross-lingual information. Instead of using a named entity tagger to identify name spelling variants, we treat names in one language as the anchor of spelling variants in another language. From sentence-aligned bilingual corpora we collect word co-occurrence statistics and calculate word translation[1] probabilities. For each source word, we group its target translations into clusters according to string edit distances, then calculate the transliteration cost between the source word and each target translation cluster. Word pairs with small transliteration costs are considered as name translations, and the target cluster contains multiple spelling variants corresponding to the source name.

We apply this approach to extract name transliteration spelling variants from bilingual corpora. We obtained tens of thousands of high precision name translation pairs. We further apply these spelling variants to Machine Translation (MT) and Information Extraction (IE) tasks, and observed statistically significant improvement on the IE task, and close to oracle improvement on the MT task.

The rest of the paper is organized as follows. In section 2 we describe the technique to identify name spelling variants from bilingual data. In section 3 and 4 we address their application to MT and IE respectively. We present our experiment results and detailed analysis in section 5. Section 6 concludes this paper with future work.

2 Finding Name Translation Variants

[1] In this paper, the translation cost measures the semantic difference between source and target names, which are estimated from their co-occurrence statistics. The transliteration cost measures their phonetic distance and are estimated based on a character transliteration model.

Starting from sentence-aligned parallel data, we run HMM alignment (Vogel et. al. 1996 & Ge 2004) to obtain a word translation model. For each source word this model generates target candidate translations as well as their translation probabilities. A typical entry is shown in Table 1. It can be observed that the Arabic name's translations include several English words with similar spellings, all of which are correct translations. However, because the lexical translation probabilities are distributed among these variants, none of them has the highest probability. As a result, the incorrect translation, *iqlim*, is assigned the highest probability and often selected in MT output. To fix this problem, it is desirable to identify and group these target spelling variants, convert them into a canonical form and merge their translation probabilities.

الخروب | Alxrwb

iqlim [0.22]	al-kharrub [0.16]	al-kharub [0.11]	overflew [0.09]
junbulat [0.05]	al-khurub [0.05]	hours [0.04]	al-kharroub [0.03]

Table 1. English translations of a Romanized Arabic name *Alxrwb* with translation probabilities.

For each source word in the word translation model, we cluster its target translations based on string edit distances using group average agglomerative clustering algorithm (Manning and Schütze, 2000). Initially each target word is a single word cluster. We calculate the average editing distance between any two clusters, and merge them if the distance is smaller than a certain threshold. This process repeats until the minimum distance between any two clusters is above a threshold. In the above example, *al-kharrub, al-kharub, al-khurub* and *al-kharroub* are grouped into a single cluster, and each of the ungrouped words remains in its single word cluster. Note that the source word may not be a name while its translations may still have similar spellings. An example is the Arabic word اعلم which is aligned to English words *brief, briefing, briefed* and *briefings*. To detect whether a source word is a name, we calculate the transliteration cost between the source word and its target translation cluster, which is defined as the average transliteration cost between the source word and each target word in the cluster. As many names are translated based on their pronunciations, the source and target names have similar phonetic features and lower transliteration costs. Word pairs whose transliteration cost is lower than an empirically selected threshold are considered as name translations.

2.1 Name Transliteration Cost

The transliteration cost measures the phonetic similarity between a source word and a target word. It is calculated based on the character transliteration model, which can be trained from bilingual name translation pairs. We segment the source and target names into characters, then run monotone[2] HMM alignment on the source and target character pairs. After the training, character transliteration probabilities can be estimated from the relevant frequencies of character alignments.

Suppose the source word f contains m characters, $f_1, f_2, \ldots, f_m$, and the target word e contains n characters, $e_1, e_2, \ldots, e_n$. For j=1, 2,…, n, letter e_j is aligned to character f_{a_j} according to the HMM aligner. Under the assumption that character alignments are independent, the word transliteration probability is calculated as

$$P(e \mid f) = \prod_{j=1}^{n} p(e_j \mid f_{a_j}) \qquad (2.1)$$

where $p(e_j \mid f_{a_j})$ is the character transliteration probability. Note that in the above configuration one target character can be aligned to only one source character, and one source character can be aligned to multiple target characters.

An example of the trained A-E character transliteration model is shown in Figure 1. The Arabic character خ is aligned with high probabilities to English letters with similar pronunciation. Because Arabic words typically omit vowels, English vowels are also aligned to Arabic characters. Given this model, the characters within a Romanized Arabic name and its English translation are aligned as shown in Figure 1.

[2] As name are typically phonetically translated, the character alignment are often monotone. There is no cross-link in character alignments.

2.2 Transliteration Unit Selection

The transliteration units are typically characters. The Arabic alphabet includes 32 characters, and the English alphbet includes 56 letters[3]. However, Chinese has about 4000 frequent characters. The imbalance of Chinese and English vocabulary sizes results in suboptimal transliteration model estimation. Each Chinese character also has a pinyin, the Romanized representation of its pronunciation. Segmenting the Chinese pinyin into sequence of Roman letters, we now have comparable vocabulary sizes for both Chinese and English. We build a pinyin transliteration model using Chinese-English name translation pairs, and compare its performance with a character transliteration model in Experiment section 5.1.

خ | x

h	K	k	a	u	i
[0.44]	[0.29]	[0.21]	[0.03]	[0.015]	[0.004]

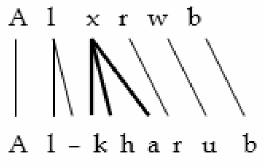


Figure 1. Example of the learned A-E character transliteration model with probabilities, and its application in the alignment between an Romanized Arabic name and an English translation.

3 Application to Machine Translation

We applied the extracted name translation spelling variants to the machine translation task. Given the name spelling variants, we updated both the translation and the language model, adding variants' probabilities to the canonical form.

Our baseline MT decoder is a phrase-based decoder as described in (Al-Onaizan and Papineni 2006). Given a source sentence, the decoder tries to find the translation hypothesis with minimum translation cost, which is defined as the log-linear combination of different feature functions, such as translation model cost, language model cost, distortion cost and

[3]Uppercase and lowercase letters plus some special symbols such as '_', '-".

sentence length cost. The translation cost includes word translation probability and phrase translation probability.

3.1 Updating The Translation Model

Given target name spelling variants $\{t_1, t_2, ..., t_m\}$ for a source name *s*, here $t_1, t_2, ..., t_m$ are sorted based on their lexical translation probabilities, $p(t_1 \mid s) \geq p(t_2 \mid s) \geq ... \geq p(t_m \mid s)$.

We select t_1 as the canonical spelling, and merge other spellings' translation probabilities with this one:

$$p(t_1 \mid s) = \sum_{j=1}^{m} p(t_m \mid s).$$

Other spelling variants get zero probability. Table 2 shows the updated word translation probabilities for "الخروب|Alxwrb". Compared with Figure 1, the translation probabilities from several spelling variants are merged with the canonical form, *al-kharrub*, which now has the highest probability in the new model.

الخروب | Alxwrb

al-kharrub [0.35]	iqlim [0.22]	al-kharub [0.0]	overflew [0.09]
junbulat [0.05]	al-khurub [0.0]	hours [0.04]	al-kharroub [0.0]

Table 2. English translations of an Arabic name الخروب|*Alxrwb* with the updated word translation model.

The phrase translation table includes source phrases, their target phrase translations and the frequencies of the bilingual phrase pair alignment. The phrase translation probabilities are calculated based on their alignment frequencies, which are collected from word aligned parallel data. To update the phrase translation table, for each phrase pair including a source name and its spelling variant in the target phrase, we replace the target name with its canonical spelling. After the mapping, two target phrases differing only in target names may end up with the identical target phrase, and their alignment frequencies are added. Phrase translation probabilities are re-estimated with the updated frequencies.

3.2 Updating The Language Model

The machine translation decoder uses a language model as a measure of a well-formedness of the output sentence. Since the updated translation model can produce only the canonical form of a group of spelling variants, the language model should be updated in that all *m*-grams ($1 \leq m \leq N$) that are spelling variants of each other are merged (and their counts added), resulting in the canonical form of the *m*-gram. Two *m*-grams are considered spelling variants of each other if they contain words t_1^i, t_2^i ($t_1^i \neq t_2^i$) at the same position *i* in the *m*-gram, and that t_1^i and t_2^i belong to the same spelling variant group.

An easy way to achieve this update is to replace every spelling variant in the original language model training data with its corresponding canonical form, and then build the language model again. However, since we do not want to replace words that are not names we need to have a mechanism for detecting names. For simplicity, in our experiments we assumed a word is a name if it is capitalized, and we replaced spelling variants with their canonical forms only for words that start with a capital letter.

4 Applying to Information Extraction

Information extraction is a crucial step toward understanding a text, as it identifies the important conceptual objects in a discourse. We address here one important and basic task of information extraction: *mention detection*[4]: we call instances of textual references to objects *mentions*, which can be either named (e.g. John Smith), nominal (the president) or pronominal (e.g. he, she). For instance, in the sentence

- President <u>*John Smith*</u> said <u>*he*</u> has no comments.

there are two mentions: *John Smith* and *he*. Similar to many classical NLP tasks, we formulate the mention detection problem as a classification problem, by assigning to each token in the text a label, indicating whether it starts a specific mention, is inside a specific mention, or is outside any mentions. Good

[4]We adopt here the ACE (NIST 2007) nomenclature.

performance in many natural language processing tasks has been shown to depend heavily on integrating many sources of information (Florian et al. 2007). We select an exponential classifier, the Maximum Entropy (MaxEnt henceforth) classifier that can integrate arbitrary types of information and make a classification decision by aggregating all information available for a given classification (Berger et al. 1996). In this paper, the MaxEnt model is trained using the *sequential conditional generalized iterative scaling* (SCGIS) technique (Goodman, 2002), and it uses a *Gaussian prior* for regularization (Chen and Rosenfeld, 2000).

In ACE, there are seven possible mention types: person, organization, location, facility, geopolitical entity (GPE), weapon, and vehicle. Experiments are run on Arabic and English. Our baseline system achieved very competitive result among systems participating in the ACE 2007 evaluation. It uses a large range of features, including lexical, syntactic, and the output of other information extraction models. These features were described in (Zitouni and Florian, 2008 & Florian et al. 2007), and are not discussed here. In this paper we focus on examining the effectiveness of name spelling variants in improving mention detection systems. We add a new feature that for each token x_i to process we fire its canonical form (class label) $C(x_i)$, representative of name spelling variants of x_i. This name spelling variant feature is also used in *conjunction* with the lexical (e.g., words and morphs in a 3-word window, prefixes and suffixes of length up to 4, stems in a 4-word window for Arabic) and syntactic (POS tags, text chunks) features.

5 Experiments

5.1 Evaluating the precision of name spelling variants

We extracted Arabic-English and English-Arabic name translation variants from sentence-aligned parallel corpora released by LDC. The accuracy of the extracted name translation spelling variants are judged by proficient Arabic and Chinese speakers.

The Arabic-English parallel corpora include 5.6M sentence pairs, 845K unique Arabic words and 403K unique English words. We trained a word translation model by running HMM alignment on the parallel data, grouped target translation with similar spellings and computed the average transliteration cost between the Arabic word and each English word in the translation clusters according to Formula 2.1. We sorted the name translation groups according to their transliteration costs, and selected 300 samples at different ranking position for evaluation (20 samples at each ranking position). The quality of the name translation variants are judged as follows: for each candidate name translation group $\{t_1, t_2, ..., t_m \mid s\}$, if the source word *s* is a name and all the target spelling variants are correct translations, it gets a credit of 1. If *s* is not a name, the credit is 0. If *s* is a name but only part of the target spelling variants are correct, it gets partial credit *n/m*, where *n* is the number of correct target translations. We evaluate only the precision of the extracted spelling variants[5]. As seen in Figure 2, the precision of the top 22K A-E name translations is 96.9%. Among them 98.5% of the Arabic words are names. The precision gets lower and lower when more non-name Arabic words are included. On average, each Arabic name has 2.47 English spelling variants, although there are some names with more than 10 spelling variants.

Switching the source and target languages, we obtained English-Arabic name spelling variants, i.e., one English name with multiple Arabic spellings. As seen in Figure 3, top 20K E-A name pairs are obtained with a precision above 87.9%, and each English name has 3.3 Arabic spellings on average. Table 3 shows some A-E and E-A name spelling variants, where Arabic words are represented in their Romanized form.

We conduct a similar experiment on the Chinese-English language pair, extracting Chinese-English and English-Chinese name spelling variants from 8.7M Chinese-English sentence pairs. After word segmentation, the Chinese vocabulary size is 1.5M words, and English vocabulary size is 1.4M words. With the

[5] Evaluating recall requires one to manually look through the space of all possible transliterations (hundreds of thousands of entries), which is impractical.

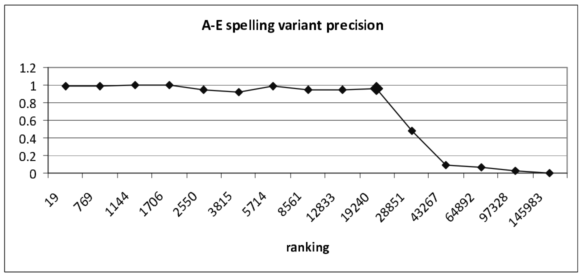


Figure 2. Arabic-English name spelling variants precision curve (Precision of evaluation sample at different ranking positions. The larger square indicates the cutoff point).

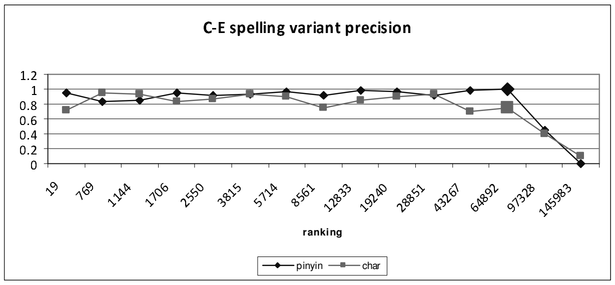


Figure 4. Chinese-English name spelling variants precision curve.

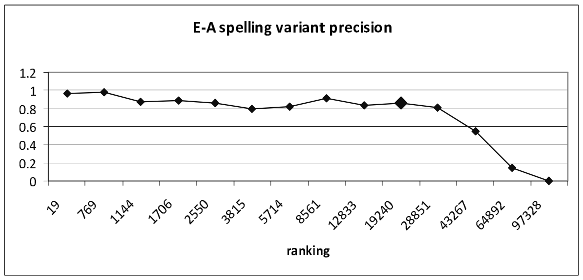


Figure 3. English-Arabic name spelling variants precision curve.

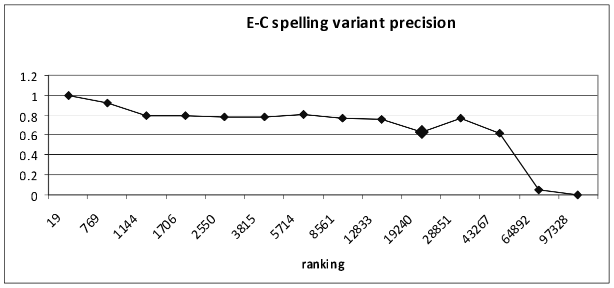


Figure 5. English-Chinese name spelling variants precision curve.

Chinese pinyin transliteration model, we extract 64K C-E name spelling variants with 93.6% precision. Figure 4 also shows the precision curve of the Chinese character transliteration model. On average the pinyin transliteration model has about 6% higher precision than the character transliteration model. The pinyin transliteration model is particularly better on the tail of the curve, extracting more C-E transliteration variants. Figure 5 shows the precision curve for E-C name spelling variants, where 20K name pairs are extracted using letter-to-character transliteration model, and obtaining a precision of 74.3%.

Table 4 shows some C-E and E-C name spelling variants. We observed errors due to word segmentation. For example, the last two Chinese words corresponding to “drenica” have additional Chinese characters, meaning “drenica region” and “drenica river”. Similarly for tenet, the last two Chinese words also have segmentation errors due to missing or spurious characters. Note that in the C-E spelling variants, the source word “韦尔曼” has 14 spelling variants. Judge solely from the spelling, it is hard to tell whether they are the same person name with different spellings.

5.2 Experiments on Machine Translation

We apply the Arabic-English name spelling variants on the machine translation task. Our baseline system is trained with 5.6M Arabic-English sentence pairs, the same training data used to extract A-E spelling variants. The language model is a modified Kneser-Ney 5-gram model trained on roughly 3.5 billion words. After pruning (using count cutoffs), it contains a total of 935 million *N*-grams. We updated the translation models and the language model with the name spelling variant class.

Table 5 shows a Romanized Arabic sentence, the translation output from the baseline system and the output from the updated models. In the baseline system output, the Arabic name “Alxrwb” was incorrectly translated into “regional”. This error was fixed in the updated model, where both translation and language models assign higher probabilities to the correct translation “al-kharroub” after spelling variant normalization.

Lang. Pair	Source Name	Target Spelling Variants
Arabic-English	Alxmyny	khomeini al-khomeini al-khomeni khomeni khomeyni *khamenei khameneh'i*
	krwby	karroubi karrubi krobi karubi karoubi kroubi
	gbryAl	gabriel gabrielle gabrial ghobrial ghybrial
English-Arabic	cirebon	syrybwn syrbwn syrbn kyrybwn bsyrybwn bsyrwbwn
	mbinda	mbyndA mbndA mbydA AmbyndA AmbAndA mbynydA
	nguyen	njwyn ngwyn ngwyyn ngyyn Angwyn nygwyyn nygwyn wnjwyn njwyyn nyjyn bnjwyn wngyyn ngwyAn njyn nykwyn

Table 3. Arabic-English and English-Arabic name spelling variant examples. *Italic words* represent different persons with similar spelling names.

Lang. Pair	Source Name	Target Spelling Variants
Chinese-English	延多维茨基 (yan/duo/wei/ci/ji)	endovitsky jendovitski yendovitski endovitski
	斯特凡尼 (si/te/fan/ni)	stefani steffani stephani stefanni stefania
	韦尔曼 (wei/er/man)	woermann wellman welman woellmann wohrmann wormann velman wollmann wehrmann verman woehrmann wellmann welmann wermann
English-Chinese	tenet	特尼特(te/ni/te) 特内特(te/nei/te) 泰内特(tai/nei/te) 特耐特(te/nai/te) 特奈特(te/nai/te) *特内特于(te/nei/te/yu) 特内(te/nei)*
	drenica	德雷尼察(de/lei/ni/cha) 德雷尼卡(de/lei/ni/ka) 特雷尼察(te/lei/ni/cha) 特雷尼查(te/lei/ni/cha) *德雷尼察区(de/lei/ni/cha/qu) 德雷尼察河 (de/lei/ni/cha/he)*
	ahmedabad	艾哈迈达巴德(ai/ha/mai/da/ba/de) 艾阿迈达巴德(ai/a/mai/da/ba/de) 艾哈默德巴德(ai/ha/mo/de/ba/de) 阿哈迈达巴德(a/ha/mai/da/ba/de)

Table 4. Chinese-English and English-Chinese name spelling variant examples with pinyin for Chinese characters. *Italic words* represent errors due to word segmentation.

Source	Alm&tmr AlAwl lAqlym *Alxrwb* AlErby AlmqAwm
Reference	the first conference of the Arab resistance in Iqlim *Kharoub*
Baseline	the first conference of the Arab *regional* resistance
Updated model	first conference of the *Al-Kharrub* the Arab resistance

Table 5. English translation output with the baseline MT system and the system with updated models

	BLEU r1n4	TER
Baseline	0.2714	51.66
Baseline+ULM+UTM	0.2718	51.46
Ref. Normalization	0.2724	51.40

Table 6. MT scores with updated TM and LM

We also evaluated the updated MT models on a MT test set. The test set includes 70 documents selected from GALE 2007 Development set. It contains 42 newswire documents and 28 weblog and newsgroup documents. There are 669 sentences with 16.3K Arabic words in the test data. MT results are evaluated against one reference human translation using BLEU (Papineni et. al. 2001) and TER (Snover et. al. 2006) scores. The results using the baseline decoder and the updated models are shown in Table 6. Applying the updated language model (ULM) and the translation model (UTM) lead to a small reduction in TER. After we apply similar name spelling normalization on the reference translation, we observed some additional improvements. Overall, the BLEU score is increased by 0.1 BLEU point and TER is reduced by 0.26.

Although the significance of correct name translation can not be fully represented by

BLEU and TER scores[6], we still want to understand the reason of the relatively small improvement. After some error analysis, we found that in the testset only 2.5% of Arabic words are names with English spelling variants. Among them, 73% name spelling errors can be corrected with the translation spelling variants obtained in section 5.1. Because the MT system is trained on the same bilingual data from which the name spelling variants are extracted, some of these Arabic names are already correctly translated in the baseline system. So the room of improvement is small. We did an oracle experiment, manually correcting the name translation errors in the first 10 documents (89 sentences with 2545 words). With only 6 name translation errors corrected, this reduced the TER from 48.83 to 48.65.

5.2 Experiments on Information Extraction

Mention detection system experiments are conducted on the ACE 2007 data sets in Arabic and English. Since the evaluation test set is not publicly available, we have split the publicly available training corpus into an 85%/15% data split. To facilitate future comparisons with work presented here, and to simulate a realistic scenario, the splits are created based on article dates: the test data is selected as the latest 15% of the data in chronological order. This way, the documents in the training and test data sets do not overlap in time, and the content of the test data is more recent than the training data. For English we use 499 documents for training and 100 documents for testing, while for Arabic we use 323 documents for training and 56 documents for testing. English and Arabic mention detection systems are using a large range of features, including lexical (e.g., words and morphs in a 3-word window, prefixes and suffixes of length up to 4, stems in a 4-word window for Arabic), syntactic (POS tags, text chunks), and the output of other information extraction models. These features were described in (Zitouni and Florian, 2008 & Florian et al. 2007) with more details. Our goal here is to investigate the effectiveness of name spelling variants information in improving mention detection system performance.

	Baseline			**Baseline+NSV**		
	P	**R**	**F**	**P**	**R**	**F**
English	84.4	80.6	**82.4**	84.6	80.9	**82.7**
Arabic	84.3	79.0	**81.6**	84.4	79.1	**81.7**

Table 7: Performance of English and Arabic mention detection systems without (Baseline) and with (Baseline+NSV) the use of name spelling variants. Performance is presented in terms of Precision (P), Recall (R), and F-measure (F).

Results in Table 7 show that the use of name spelling variants (**NSV**) improves mention detection systems performance, especially for English; an interesting improvement is obtained in recall – which is to be expected, given the method –, but also in precision, leading to systems with better performance in terms of F-measure (82.4 vs. 82.7). This improvement in performance is statistically significant according to the stratified bootstrap re-sampling approach (Noreen 1989). This approach is used in the named entity recognition shared task of CoNLL-2002[7]. However, the small improvement obtained for Arabic is not statistically significant based on the approach described earlier. One hypothesis is that Arabic name spelling variants are not rich enough and that a better tuning of the alignment score is required to improve precision.

6 Conclusion

We proposed a cross-lingual name spelling variants extraction technique. We extracted tens of thousands of high precision bilingual name translation spelling variants. We applied the spelling variants to the IE task, observing statistically significant improvements over a strong baseline system. We also applied the spelling variants to MT task and even though the overall improvement is relatively small, it achieves performance close to the one observed in an oracle experiment.

[6] These scores treat information bearing words, like names, the same as any other words, like punctuations.

[7] http://www.cnts.ua.ac.be/conll2002/ner/

Acknowledgments

This work was supported by DARPA/IPTO Contract No. HR0011-06-2-0001 under the GALE program. We are grateful to Yaser Al-Onaizan, Salim Roukos and anonymous reviewers for their constructive comments.

References

Al-Onaizan, Y. and Papineni, K. Distortion Models for Statistical Machine Translation. In Proceedings of the 44th Annual Meeting on Association For Computational Linguistics. Sydney, Australia. July 2006.

Al-Onaizan, Y. and Knight, K. Translating named entities using monolingual and bilingual resources. In *Proceedings of the 40th Annual Meeting on Association For Computational Linguistics* (Philadelphia, Pennsylvania, July 07 - 12, 2002). Annual Meeting of the ACL. Association for Computational Linguistics, Morristown, NJ. 2002

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

Bhagat, R. and Hovy, E. "Phonetic Models for Generating Spelling Variants", *In Proceedings International Joint Conference of Artificial Intelligence* (IJCAI). Hyderabad, India. 2007.

Chen, S. and Rosenfeld R. A survey of smoothing techniquesfor ME models. IEEE Trans. On Speech and Audio Processing. 2002

Florian, R., Hassan, H., Ittycheriah, A., Jing, H., Kambhatla, N., Luo, X., Nicolov, N., and Roukos. S. A statistical model for multilingual entity detection and tracking. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*: HLT-NAACL 2004, pages 1–8.

Ge, N. Improvements in Word Alignments. Presentation given at DARPA/TIDES NIST MT Evaluation workshop. 2004

Goodman. J. Sequential conditional generalized iterative scaling. In *Proceedings of the 40th Annual Meeting on Association For Computational Linguistics* (Philadelphia, Pennsylvania, July 07 - 12, 2002). Annual Meeting of the ACL. Association for Computational Linguistics, Morristown, NJ. 2002

Huang, F., Vogel, S., and Waibel, A. Automatic extraction of named entity translingual equivalence based on multi-feature cost minimization. In *Proceedings of the ACL 2003 Workshop on Multilingual and Mixed-Language Named Entity Recognition* - Annual Meeting of the ACL. Association for Computational Linguistics, Morristown, NJ, 2003

Ji, H. and Grishman. R. Collaborative Entity Extraction and Translation. *Proc. International Conference on Recent Advances in Natural Language Processing*. Borovets, Bulgaria. Sept 2007.

Knuth, D. *The Art of Computer Programming – Volume 3: Sorting and Searching*. Addison- Wesley Publishing Company, 1973.

Linden, K. "Multilingual Modeling of Cross-Lingual Spelling Variants", *Information Retrieval*, Vol. 9, No. 3. (June 2006), pp. 295-310.

Manning, C.D., and Schutze., H. *Foundations of Statistical Natural Language Processing*. MIT Press, 2000

NIST. 2007. The ACE evaluation plan. www.nist.gov/speech/tests/ace/index.htm.

Noreen, E. W. *Computer-Intensive Methods for Testing Hypothesis*. John Wiley Sons. 1989

Papineni, K.A., Roukos, S., Ward, T., Zhu, W.J. BLEU: a method for automatic evaluation of machine translation. *Technical Report RC22176 (W0109-022)*, IBM Research Division, Thomas J. Watson Research Center (2001)

Raghavan, H. and Allan, J., "Matching Inconsistently Spelled Names in Automatic Speech Recognizer Output for Information Retrieval*," the Proceedings of HLT/EMNLP* 2005, pp. 451-458.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. "A Study of Translation Edit Rate with Targeted Human Annotation," *Proceedings of Association for Machine Translation in the Americas*, 2006.

Toivonen, J., Pirkola, A., Keskustalo, H., Visala, K. and Järvelin, K. Translating cross-lingual spelling variants using transformation rules. *Inf. Process. Manage.* 41(4): 859-872 (2005)

Vogel, S., Ney, H., and Tillmann, C.. HMM-based word alignment in statistical translation. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 2* (Copenhagen, Denmark, August 05 - 09, 1996). International Conference On Computational Linguistics. Association for Computational Linguistics, Morristown, NJ 1996

Zitouni, I., Florian R.. Mention Detection Crossing the Language Barrier. Proceedings of Conference on Empirical Methods in Natural Language Processing. Waikiki, Honolulu, Hawaii (October, 2008)

Zitouni, I., Sorensen, J., Luo, X., and Florian, R. The impact of morphological stemming on Arabic mention detection and coreference resolution. In Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages. The 43rd Annual Meeting of the Association for Computational Linguistics. Ann Arbor (June, 2005)

A Dependency-based Word Subsequence Kernel

Rohit J. Kate
Department of Computer Sciences
The University of Texas at Austin
1 University Station C0500
Austin, TX 78712-0233, USA
rjkate@cs.utexas.edu

Abstract

This paper introduces a new kernel which computes similarity between two natural language sentences as the number of paths shared by their dependency trees. The paper gives a very efficient algorithm to compute it. This kernel is also an improvement over the word subsequence kernel because it only counts linguistically meaningful word subsequences which are based on word dependencies. It overcomes some of the difficulties encountered by syntactic tree kernels as well. Experimental results demonstrate the advantage of this kernel over word subsequence and syntactic tree kernels.

1 Introduction

Kernel-based learning methods (Vapnik, 1998) are becoming increasingly popular in natural language processing (NLP) because they allow one to work with potentially infinite number of features without explicitly constructing or manipulating them. In most NLP problems, the data is present in structured forms, like strings or trees, and this structural information can be effectively passed to a kernel-based learning algorithm using an appropriate kernel, like a string kernel (Lodhi et al., 2002) or a tree kernel (Collins and Duffy, 2001). In contrast, feature-based methods require reducing the data to a pre-defined set of features often leading to some loss of the useful structural information present in the data.

A kernel is a measure of similarity between every pair of examples in the data and a kernel-based machine learning algorithm accesses the data only through these kernel values. For example, the string kernel (Lodhi et al., 2002; Cancedda et al., 2003) computes the similarity between two natural language strings as the number of common word subsequences between them. A subsequence allows gaps between the common words which are penalized according to a parameter. Each word subsequence hence becomes an implicit feature used by the kernel-based machine learning algorithm. A problem with this kernel is that many of these word subsequences common between two strings may not be semantically expressive or linguistically meaningful[1]. Another problem with this kernel is that if there are long-range dependencies between the words in a common word subsequence, then they will unfairly get heavily penalized because of the presence of word gaps.

The syntactic tree kernel presented in (Collins and Duffy, 2001) captures the structural similarity between two syntactic trees as the number of syntactic subtrees common between them. However, often syntactic parse trees may share syntactic subtrees which correspond to very different semantics based on what words they represent in the sentence. On the other hand, some subtrees may differ syntactically but may represent similar underlying semantics. These differences can become particularly problematic if the tree kernel is to be used for tasks which require semantic processing.

This paper presents a new kernel which computes similarity between two sentences as the the number of paths common between their dependency trees.

[1](Lodhi et al., 2002) use character subsequences instead of word subsequences which are even less meaningful.

(a) *A fat cat was chased by a dog.*
(b) *A cat with a red collar was chased two days ago by a fat dog.*

Figure 1: Two natural language sentences.

It improves over the word subsequence kernel because it only counts the word subsequences which are linked by dependencies. It also circumvents some of the difficulties encountered with the syntactic tree kernel when applied for semantic processing tasks.

Although several dependency-tree-based kernels and modifications to syntactic tree kernels have been proposed which we briefly discuss in the Related Work section, to our best knowledge no previous work has presented a kernel based on dependency paths which offers some unique advantages. We also give a very efficient algorithm to compute this kernel. We present experimental results on the task of domain-specific semantic parsing demonstrating the advantage of this kernel over word subsequence and syntactic tree kernels.

The following section gives some background on string and tree kernels. Section 3 then introduces the dependency-based word subsequence kernel and gives an efficient algorithm to compute it. Some of the related work is discussed next, followed by experiments, future work and conclusions.

2 String and Tree Kernels

2.1 Word-Subsequence Kernel

A kernel between two sentences measures the similarity between them. Lodhi et al. (2002) presented a string kernel which measures the similarity between two sentences, or two documents in general, as the number of character subsequences shared between them. This was extended by Cancedda et al. (2003) to the number of common word subsequences between them. We will refer to this kernel as the *word subsequence kernel*.

Consider the two sentences shown in Figure 1. Some common word subsequences between them are "a cat", "was chased by", "by a dog", "a cat chased by a dog", etc. Note that the subsequence "was chased by" is present in the second sentence but it requires skipping the words "two days ago" or has a gap of three words present in it. The kernel downweights the presence of gaps by a decay factor $\lambda \epsilon (0, 1]$. If g_1 and g_2 are the sum totals of gaps for a subsequence present in the two sentences respectively, then the contribution of this subsequence towards the kernel value will be $\lambda^{g_1+g_2}$. The kernel can be normalized to have values in the range of $[0, 1]$ to remove any bias due to different sentence lengths. Lodhi et al. (2002) give a dynamic programming algorithm to compute string subsequence kernels in $O(nst)$ time where s and t are the lengths of the two input strings and n is the maximum length of common subsequences one wants to consider. Rousu and Shawe-Taylor (2005) present an improved algorithm which works faster when the vocabulary size is large. Subsequence kernels have been used with success in NLP for text classification (Lodhi et al., 2002; Cancedda et al., 2003), information extraction (Bunescu and Mooney, 2005b) and semantic parsing (Kate and Mooney, 2006).

There are, however, some shortcomings of this word subsequence kernel as a measure of similarity between two sentences. Firstly, since it considers all possible common subsequences, it is not sensitive to whether the subsequence is linguistically meaningful or not. For example, the meaningless subsequences "cat was by" and "a was a" will also be considered common between the two sentences by this kernel. Since these subsequences will be used as implicit features by the kernel-based machine learning algorithm, their presence can only hurt the performance. Secondly, if there are long distance dependencies between the words of the subsequence present in a sentence then the subsequence will get unfairly penalized. For example, the most important word subsequence shared between the two sentences shown in Figure 1 is "a cat was chased by a dog" which will get penalized by total gap of eight words coming from the second sentence and a gap of one word from the first sentence. Finally, the kernel is not sensitive to the relations between the words, for example, the kernel will consider "a fat dog" as a common subsequence although in the first sentence "a fat" relates to the cat and not to the dog.

2.2 Syntactic Tree Kernel

Syntactic tree kernels were first introduced by Collins and Duffy (2001) and were also used by

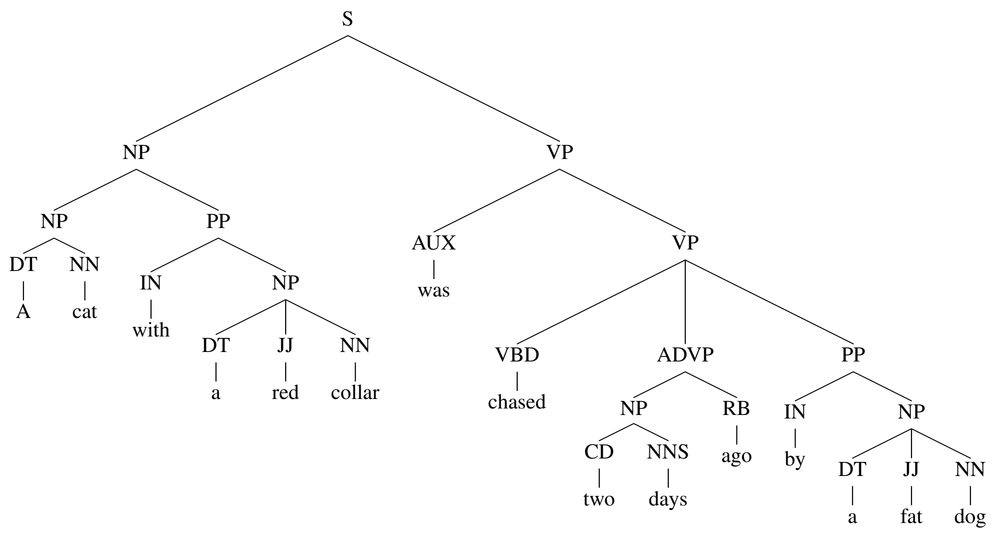


Figure 3: Syntactic parse tree of the sentence shown in Figure 1 (b).

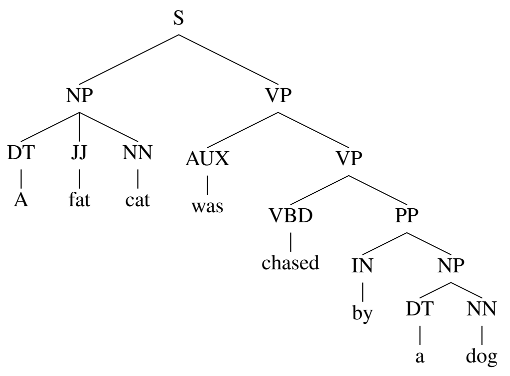


Figure 2: Syntactic parse tree of the sentence shown in Figure 1 (a).

Collins (2002) for the task of re-ranking syntactic parse trees. They define a kernel between two trees as the number of subtrees shared between them. A subtree is defined as any subgraph of the tree which includes more than one node, with the restriction that entire productions must be included at every node. The kernel defined this way captures most of the structural information present in the syntactic parse trees in the form of tree fragments which the kernelized learning algorithms can then implicitly use as features. The kernel can be computed in $O(|N_1||N_2|)$ time, where $|N_1|$ and $|N_2|$ are the number of nodes of the two trees. An efficient algorithm to compute tree kernels was given by Moschitti (2006a) which runs in close to linear time in the size of the input trees.

One drawback of this tree kernel, though, particularly when used for any task requiring semantic processing, is that it may match syntactic subtrees between two trees even though they represent very dissimilar things in the sentence. For example, between the syntactic parse trees shown in Figures 2 and 3 for the two sentences shown in Figure 1, the syntactic tree kernel will find (NP (DT a) JJ NN) as a common subtree but in the first sentence it represents "cat" while in the second it represents "collar" and "dog". It will also find "(NP (DT a) (JJ fat) NN)" as a common subtree which again refers to "cat" in the first sentence and "dog" in the second sentence. As another example, consider two simple sentences: (S (NP Chip) (VP (V saw) (NP Dale))) and (S (NP Mary) (VP (V heard) (NP Sally))). Even though semantically nothing is similar between them, the syntactic tree kernel will still find common subtrees (S NP VP), (VP N NP) and (S NP (VP V NP)). The underlying problem is that the syntactic tree kernel tends to overlook the words of the sentences which, in fact, carry the essential semantics. On the other hand, although (NP (DT a) (NN cat)) and (NP (DT a) (JJ fat) (NN cat)) represent very similar concepts but the kernel will not capture this high level similarity between the two constituents, and will only find (DT a) and (NN cat) as the common substructures. Finally, the most important similarity between the two sentences is "a cat was chased by a dog" which will not be captured by this kernel because

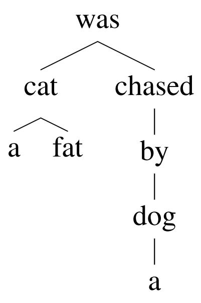


Figure 4: Dependency tree of the sentence shown in Figure 1 (a).

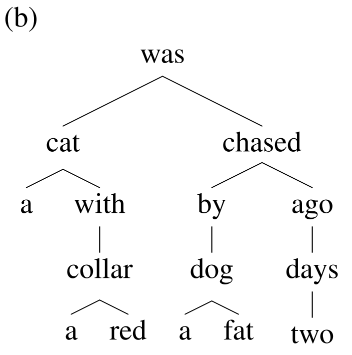


Figure 5: Dependency tree of the sentence shown in Figure 1 (b).

there is no common subtree which covers it. The Related Work section discusses some modifications that have been proposed to the syntactic tree kernel.

3 A Dependency-based Word Subsequence Kernel

A dependency tree encodes functional relationships between the words in a sentence (Hudson, 1984). The words of the sentence are the nodes and if a word complements or modifies another word then there is a child to parent edge from the first word to the second word. Every word in a dependency tree has exactly one parent except for the root word. Figures 4 and 5 show dependency trees for the two sentences shown in Figure 1. There has been a lot of progress in learning dependency tree parsers (McDonald et al., 2005; Koo et al., 2008; Wang et al., 2008). They can also be obtained indirectly from syntactic parse trees utilizing the head words of the constituents.

We introduce a new kernel which takes the words into account like the word-subsequence kernel and also takes the syntactic relations between them into account like the syntactic tree kernel, however, it does not have the shortcomings of the two kernels pointed out in the previous section. This kernel counts the number of common paths between the dependency trees of the two sentences. Another way to look at this kernel is that it counts all the common word subsequences which are linked by dependencies. Hence we will call it a *dependency-based word subsequence kernel*. Since the implicit features it uses are dependency paths which are enumerable, it is a well defined kernel. In other words, an example gets implicitly mapped to the feature space in which each dependency path is a dimension.

The dependency-based word subsequence kernel will find the common paths 'a $\rightarrow$ cat', 'cat $\rightarrow$ was $\leftarrow$ chased', 'chased $\leftarrow$ by $\leftarrow$ dog' among many others between the dependency trees shown in Figures 4 and 5. The arrows are always shown from child node to the parent node. A common path takes into account the direction between the words as well. Also note that it will find the important subsequence 'a $\rightarrow$ cat $\rightarrow$ was $\leftarrow$ chased $\leftarrow$ by $\leftarrow$ dog $\leftarrow$ a' as a common path.

It can be seen that the word subsequences this kernel considers as common paths are linguistically meaningful. It is also not affected by long-range dependencies between words because those words are always directly linked in a dependency tree. There is no need to allow gaps in this kernel either because related words are always linked. It also won't find 'a fat' as a common path because in the first tree "cat" is between the two words and in the second sentence "dog" is between them. Thus it does not have the shortcomings of the word subsequence kernel. It also avoids the shortcomings of the syntactic tree kernel because the common paths are words themselves and syntactic labels do not interfere in capturing the similarity between the two sentences. It will not find anything common between dependency trees for the sentences "Chip saw Dale" and "Mary heard Sally". But it will find 'a $\rightarrow$ cat' as a common path between "a cat" and "a fat cat". We however note that this kernel does not use general syntactic categories, unlike the syntactic tree kernel, which will limit its applicability to the tasks which depend on the syntactic categories, like re-ranking syntactic parse trees.

We now give an efficient algorithm to compute all the common paths between two trees. To our best knowledge, no previous work has considered this problem. The key observation for this algorithm is that a path in a tree always has a structure in which nodes (possibly none) go up to a highest node followed by nodes (possibly none) coming down. Based on this observation we compute two quantities for every pair of nodes between the two trees. We call the first quantity *common downward paths* (CDP) between two nodes, one from each tree, and it counts the number of common paths between the two trees which originate from those two nodes and *which always go downward*. For example, the common downward paths between the 'chased' node of the tree in Figure 4 and the 'chased' node of the tree in Figure 5 are 'chased $\leftarrow$ by', 'chased $\leftarrow$ by $\leftarrow$ dog' and 'chased $\leftarrow$ by $\leftarrow$ dog $\leftarrow$ a'. Hence $CDP(chased, chased)$ = 3. A word may occur multiple times in a sentence so the CDP values will be computed separately for each occurrence. We will shortly give a fast recursive algorithm to compute CDP values.

Once these CDP values are known, using these the second quantity is computed which we call *common peak paths* (CPP) between every two nodes, one from each tree. This counts the number of common paths between the two trees which *peak* at those two nodes, i.e. these nodes are the highest nodes in those paths. For example, 'was' is the peak for the path 'a $\rightarrow$ cat $\rightarrow$ was $\leftarrow$ chased'. Since every common path between the two trees has a unique highest node, once these CPP values have been computed, the number of common paths between the two trees is simply the sum of all these CPP values.

We now describe how all these values are efficiently computed. The CDP values between every two nodes n_1 and n_2 of the trees T_1 and T_2 respectively, is recursively computed as follows:

$$CDP(n_1, n_2) = 0 \; if \; n_1.w \neq n_2.w$$

otherwise,

$$CDP(n_1, n_2) = \sum_{\substack{c_1 \epsilon C(n_1)\\ c_2 \epsilon C(n_2)\\ c_1.w = c_2.w}} (1 + CDP(c_1, c_2))$$

In the first equation, $n.w$ stands for the word at the node n. If the words are not equal then there cannot be any common downward paths originating from the nodes. In the second equation, $C(n)$ represents the set of children nodes of the node n in a tree. If the words at two children nodes are the same, then the number of common downward paths from the parent will include all the common downward paths at the two children nodes incremented with the link from the parent to the children. In addition the path from parent to the child node is also a common downward path. For example, in the trees shown in Figures 4 and 5, the nodes with word 'was' have 'chased' as a common child. Hence all the common downward paths originating from 'chased' (namely 'chased $\leftarrow$ by', 'chased $\leftarrow$ by $\leftarrow$ dog' and 'chased $\leftarrow$ by $\leftarrow$ dog $\leftarrow$ a') when incremented with 'was $\leftarrow$ chased' become common downward paths originating from 'was'. In addition, the path 'was $\leftarrow$ chased' itself is a common downward path. Since 'cat' is also a common child at 'was', it's common downward paths will also be added.

The CDP values thus computed are then used to compute the CPP values as follows:

$$CPP(n_1, n_2) = 0 \; if \; n_1.w \neq n_2.w$$

otherwise,

$$CPP(n_1, n_2) = CDP(n_1, n_2) + \sum_{\substack{c_1, \hat{c_1} \epsilon C(n_1)\\ c_2, \hat{c_2} \epsilon C(n_2)\\ c_1.w = c_2.w\\ \hat{c_1}.w = \hat{c_2}.w}} \left(\begin{array}{c} 1 + CDP(c_1, c2) + CDP(\hat{c_1}, \hat{c_2}) + \\ CDP(c_1, c_2) * CDP(\hat{c_1}, \hat{c_2}) \end{array} \right)$$

If the two nodes are not equal then the number of common paths that peak at them will be zero. If the nodes are equal, then all the common downward paths between them will also be the paths that peak at them, hence it is the first term in the above equation. Next, the remaining paths that peak at them can be counted by considering every pair of common children nodes represented by c_1 & c_2 and $\hat{c_1}$ & $\hat{c_2}$. For example, for the common node 'was' in Figures 4 and 5, the children nodes 'cat' and 'chased' are common. The path 'cat $\rightarrow$ was $\leftarrow$ chased' is a path that peaks at 'was', hence 1 is added in the second term. All the downward paths from ‘cat’ when incremented up to ‘was’ and down to ‘chased’ are also the paths that peak at ‘was’ (namely ‘a → cat → was ← chased’). Similarly, all the downward paths from ‘chased’ when incremented up to ‘was’ and down to ‘cat’ are also paths that peak at ‘was’ (‘cat → was ← chased ← by’, ‘cat → was ← chased ← by ← dog’, etc.). Hence the next two terms are present in the equation. Finally, all the downward paths from ‘cat’ when incremented up to ‘was’ and down to every downward path from ‘chased’ are also the paths that peak at ‘was’ (‘a → cat → was ← chased ← by’, ‘a → cat → was ← chased ← by ← dog’ etc.). Hence there is the product term present in the equation. It is important not to re-count a path from the opposite direction hence the two pairs of common children are considered only once (i.e. not reconsidered symmetrically).

The dependency word subsequence kernel between two dependency trees T_1 and T_2 is then simply:

$$K(T_1, T_2) = \sum_{\substack{n_1 \epsilon T_1 \\ n_2 \epsilon T_2 \\ n_1.w = n_2.w}} (1 + CPP(n_1, n_2))$$

We also want to count the number of common words between the two trees in addition to the number of common paths, hence 1 is added in the equation. The kernel is normalized to remove any bias due to different tree sizes:

$$K_{normalized}(T_1, T_2) = \frac{K(T_1, T_2)}{\sqrt{K(T_1, T_1) * K(T_2, T_2)}}$$

Since for any long path common between two trees, there will be many shorter paths within it which will be also common between the two trees, it is reasonable to downweight the contribution of long paths. We do this by introducing a parameter $\alpha \epsilon (0, 1]$ and by downweighting a path of length l by α^l. A similar mechanism was also used in the syntactic tree kernel (Collins and Duffy, 2001).

The equations for computing CDP and CPP are accordingly modified as follows to accommodate this downweighting.

$$CDP(n_1, n_2) = 0 \; if \; n_1.w \neq n_2.w$$

otherwise,

$$CDP(n_1, n_2) = \sum_{\substack{c_1 \epsilon C(n_1) \\ c_2 \epsilon C(n_2) \\ c_1.w = c_2.w}} (\alpha + \alpha * CDP(c_1, c_2))$$

$$CPP(n_1, n_2) = 0 \; if \; n_1.w \neq n_2.w$$

otherwise,

$$CPP(n_1, n_2) = CDP(n_1, n_2) + \sum_{\substack{c_1, \hat{c_1} \epsilon C(n_1) \\ c_2, \hat{c_2} \epsilon C(n_2) \\ c_1.w = c_2.w \\ \hat{c_1}.w = \hat{c_2}.w}} \left(\begin{array}{c} \alpha^2 + \alpha * CDP(c_1, c2) + \\ \alpha * CDP(\hat{c_1}, \hat{c_2}) + \\ \alpha^2 * CDP(c_1, c_2) * CDP(\hat{c_1}, \hat{c_2}) \end{array} \right)$$

This algorithm to compute all the common paths between two trees has worst time complexity of $O(|T_1||T_2|)$, where $|T_1|$ and $|T_2|$ are the number of nodes of the two trees T_1 and T_2 respectively. This is because CDP computations are needed for every pairs of nodes between the two trees and is recursively computed. Using dynamic programming their recomputations can be easily avoided. The CPP computations then simply add the CDP values[2]. If the nodes common between the two trees are sparse then the algorithm will run much faster. Since the algorithm only needs to store the CDP values, its space complexity is $O(|T_1||T_2|)$. Also note that this algorithm computes the number of common paths of *all* lengths unlike the word subsequence kernel in which the maximum subsequence length needs to be specified and the time complexity then depends on this length.

4 Related Work

Several modifications to the syntactic tree kernels have been proposed to overcome the type of problems pointed out in Subsection 2.2. Zhang et al. (2007) proposed a grammar-driven syntactic tree kernel which allows soft matching between the subtrees of the trees if that is deemed appropriate by the grammar. For example, their kernel will be able

[2]This analysis uses the fact that any node in a tree on average has $O(1)$ number of children.

to match the subtrees (NP (DT a) (NN cat)) and (NP (DT a) (JJ fat) (NN cat)) with some penalty. Moschitti (2006b) proposed a partial tree kernel which can partially match subtrees. Moschitti et al. (2007) proposed a tree kernel over predicate-argument structures of sentences based on the PropBank labels. Che et al. (2006) presented a hybrid tree kernel which combines a constituent and a path kernel. We however note that the paths in this kernel link predicates and their arguments and are very different from general paths in a tree that our dependency-based word subsequence kernel uses. Shen et al. (2003) proposed a lexicalized syntactic tree kernel which utilizes LTAG-based features. Toutanova et al. (2004) compute similarity between two HPSG parse trees by finding similarity between the leaf projection paths using string kernels.

A few kernels based on dependency trees have also been proposed. Zelenko et al. (2003) proposed a tree kernel over shallow parse tree representations of sentences. This tree kernel was slightly generalized by Culotta and Sorensen (2004) to compute similarity between two dependency trees. In addition to the words, this kernel also incorporates word classes into the kernel. The kernel is based on counting matching subsequences of children of matching nodes. But as was also noted in (Bunescu and Mooney, 2005a), this kernel is *opaque* i.e. it is not obvious what the implicit features are and the authors do not describe it either. In contrast, our dependency-based word subsequence kernel, which also computes similarity between two dependency trees, is very transparent with the implicit features being simply the dependency paths. Their kernel is also very time consuming and in their more general sparse setting it requires $O(mn^3)$ time and $O(mn^2)$ space, where m and n are the number of nodes of the two trees ($m >= n$) (Zelenko et al., 2003).

Bunescu and Mooney (2005a) give a shortest path dependency kernel for relation extraction. Their kernel, however, does not find similarity between two sentences but between the shortest dependency paths connecting the two entities of interests in the sentences. This kernel uses general dependency graphs but if the graph is a tree then the shortest path is the only path between the entities. Their kernel also uses word classes in addition to the words themselves.

5 Experiments

We show that the new dependency-based word subsequence kernel performs better than word subsequence kernel and syntactic tree kernel on the task of domain-specific semantic parsing.

5.1 Semantic Parsing

Semantic parsing is the task of converting natural language sentences into their domain-specific complete formal meaning representations which an application can execute, for example, to answer database queries or to control a robot. A learning system for semantic parsing induces a semantic parser from the training data of natural language sentences paired with their respective meaning representations. KRISP (Kate and Mooney, 2006) is a semantic parser learning system which uses word subsequence kernel based SVM (Cristianini and Shawe-Taylor, 2000) classifiers and was shown to be robust to noise compared to other semantic parser learners. The system learns an SVM classifier for every production of the meaning representation grammar which tells the probability with which a substring of the sentence represents the semantic concept of the production. Using these classifiers a complete meaning representation of an input sentence is obtained by finding the most probable parse which covers the whole sentence. For details please refer to (Kate and Mooney, 2006).

The key operation in KRISP is to find the similarity between any two substrings of two natural language sentences. Word subsequence kernel was employed in (Kate and Mooney, 2006) to compute the similarity between two substrings. We modified KRISP so that the similarity between two substrings can also be computed using the syntactic tree kernel and the dependency-based word subsequence kernel. For applying the syntactic tree kernel, the syntactic subtree over a substring of a sentence is determined from the syntactic tree of the sentence by finding the lowest common ancestor of the words in this substring and then considering the smallest subtree rooted at this node which includes all the words of the substring. For applying the dependency-based word subsequence kernel to two substrings of a sentence, the kernel computation was suitably modified so that the common paths between the two dependency trees always begin and end with the words present in the substrings. This is achieved by including only those downward paths in computations of CDP which end with words within the given substrings. These paths relate the words within the substrings perhaps using words outside of these substrings.

5.2 Methodology

We measure the performance of KRISP obtained using the three types of kernels on the GEOQUERY corpus which has been used previously by several semantic parsing learning systems. It contains 880 natural language questions about the US geography paired with their executable meaning representations in a functional query language (Kate et al., 2005). Since the purpose of the experiments is to compare different kernels and not different semantic parsers, we do not compare the performance with other semantic parser learning systems. The training and testing was done using standard 10-fold cross-validation and the performance was measured in terms of precision (the percentage of generated meaning representations that were correct) and recall (the percentage of all sentences for which correct meaning representations were obtained). Since KRISP assigns confidences to the meaning representations it outputs, an entire range of precision-recall trade-off can be obtained. We measure the best F-measure (harmonic mean of precision and recall) obtained when the system is trained using increasing amounts of training data.

Since we were not interested in the accuracy of dependency trees or syntactic trees but in the comparison between various kernels, we worked with gold-standard syntactic trees. We did not have gold-standard dependency trees available for this corpus so we obtained them indirectly from the gold-standard syntactic trees using the head-rules from (Collins, 1999). We however note that accurate syntactic trees can be obtained by training a syntactic parser on WSJ treebank and gold-standard parse trees of some domain-specific sentences (Kate et al., 2005).

In the experiments, the α parameter of the dependency-based word subsequence kernel was set to 0.25, the λ parameter of the word subsequence kernel was fixed to 0.75 and the downweighting parameter for the syntactic tree kernel was fixed to 0.4. These were determined through pilot experiments with a smaller portion of the data set. The maximum length of subsequences required by the word subsequence kernel was fixed to 3, a longer length was not found to improve the performance and was only increasing the running time.

Examples	Dependency	Word	Syntactic
40	**25.62**	21.51	23.65
80	**45.30**	42.77	43.14
160	**63.78**	61.22	59.66
320	**72.44**	70.36	67.05
640	77.32	77.82	74.26
792	79.79	79.09	76.62

Table 1: Results on the semantic parsing task with increasing number of training examples using dependency-based word subsequence kernel, word subsequence kernel and syntactic tree kernel.

5.3 Results

Table 1 shows the results. The dependency-based word subsequence kernel always performs better than the syntactic tree kernel. All the numbers under the dependency kernel were found statistically significant ($p < 0.05$) over the corresponding numbers under the syntactic tree kernel based on paired t-tests. The improvement of the dependency-based word subsequence kernel over the word subsequence kernel is greater with less training data, showing that the dependency information is more useful when the training data is limited. The performance converges with higher amounts of training data. The numbers shown in bold were found statistically significant over the corresponding numbers under the word subsequence kernel.

It may be noted that syntactic tree kernel is mostly doing worse than the word subsequence kernel. We believe this is because of the shortcomings of the syntactic tree kernel pointed out in Subsection 2.2. Since this is a semantic processing task, the words play an important role and the generalized syntactic categories are not very helpful.

6 Future Work

In future, the dependency-based word subsequence kernel could be extended to incorporate word classes like the kernels presented in (Bunescu and Mooney, 2005a; Zelenko et al., 2003). It should be possible to achieve this by incorporating matches between word classes in addition to the exact word matches in the kernel computations similar to the way in which the word subsequence kernel was extended to incorporate word classes in (Bunescu and Mooney, 2005b). This will generalize the kernel and make it more robust to data sparsity.

The dependency-based word subsequence kernel could be tested on other tasks which require computing similarity between sentences or texts, like text classification, paraphrasing, summarization etc. We believe this kernel will help improve performance on those tasks.

7 Conclusions

We introduced a new kernel which finds similarity between two sentences as the number of common paths shared between their dependency trees. This kernel can also be looked upon as an improved word subsequence kernels which only counts the common word subsequences which are related by dependencies. We also gave an efficient algorithm to compute this kernel. The kernel was shown to out-perform the word subsequence kernel and the syntactic tree kernel on the task of semantic parsing.

References

Razvan C. Bunescu and Raymond J. Mooney. 2005a. A shortest path dependency kernel for relation extraction. In *Proc. of HLT/EMNLP-05*, pages 724–731, Vancouver, BC, October.

Razvan C. Bunescu and Raymond J. Mooney. 2005b. Subsequence kernels for relation extraction. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, Vancouver, BC.

Nicola Cancedda, Eric Gaussier, Cyril Goutte, and Jean Michel Renders. 2003. Word sequence kernels. *Journal of Machine Learning Research, Special Issue on Machine Learning Methods for Text and Images*, 3:1059–1082, February.

Wanxiang Che, Min Zhang, Ting Liu, and Sheng Li. 2006. A hybrid convolution tree kernel for semantic role labeling. In *Proc. of COLING/ACL-06*, pages 73–80, Sydney, Australia, July.

Michael Collins and Nigel Duffy. 2001. Convolution kernels for natural language. In *Proc. of NIPS-2001*.

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

Michael Collins. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proc. of ACL-2002*, pages 263–270, Philadelphia, PA, July.

Nello Cristianini and John Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proc. of ACL-04*, pages 423–429, Barcelona, Spain, July.

Richard Hudson. 1984. *Word Grammar*. Blackwell.

Rohit J. Kate and Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proc. of COLING/ACl-06*, pages 913–920, Sydney, Australia, July.

Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to transform natural to formal languages. In *Proc. AAAI-2005*, pages 1062–1068, Pittsburgh, PA, July.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proc. of ACL-08*, pages 595–603, Columbus, Ohio, June.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of HLT/EMNLP-05*, pages 523–530, Vancouver, BC.

Alessandro Moschitti, Silvia Quarteroni, Roberto Basili, and Suresh Manandhar. 2007. Exploiting syntactic and shallow semantic kernels for question answer classification. In *Proc. of ACL-07*, pages 776–783, Prague, Czech Republic, June.

Alessandro Moschitti. 2006a. Making tree kernels practical for natural language learning. In *Proc. of EACL-06*, pages 113–120, Trento, Italy, April.

Alessandro Moschitti. 2006b. Syntactic kernels for natural language learning: the semantic role labeling case. In *Proc. of HLT/NAACL-06, short papers*, pages 97–100, New York City, USA, June.

Juho Rousu and John Shawe-Taylor. 2005. Efficient computation of gapped substring kernels on large alphabets. *Journal of Machine Learning Research*, 6:1323–1344.

Libin Shen, Anoop Sarkar, and Aravind Joshi. 2003. Using ltag based features in parse reranking. In *Proc. of EMNLP-2003*, pages 89–96, Sapporo, Japan, July.

Kristina Toutanova, Penka Markova, and Christopher Manning. 2004. The leaf projection path view of parse trees: Exploring string kernels for HPSG parse selection. In *Proc. EMNLP-04*, pages 166–173, Barcelona, Spain, July.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. John Wiley & Sons.

Qin Iris Wang, Dale Schuurmans, and Dekang Lin. 2008. Semi-supervised convex training for dependency parsing. In *Proceedings of ACL-08: HLT*, pages 532–540, Columbus, Ohio, June.

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

Min Zhang, Wanxiang Che, Aiti Aw, Chew Lim Tan, Guodong Zhou, Ting Liu, and Sheng Li. 2007. A grammar-driven convolution tree kernel for semantic role classification. In *Proc. of ACL-2007*, pages 200–207, Prague, Czech Republic, June.

Bridging Lexical Gaps between Queries and Questions on Large Online Q&A Collections with Compact Translation Models

Jung-Tae Lee[†] and **Sang-Bum Kim**[§] and **Young-In Song**[‡] and **Hae-Chang Rim**[†]
[†]Dept. of Computer & Radio Communications Engineering, Korea University, Seoul, Korea
[§]Search Business Team, SK Telecom, Seoul, Korea
[‡]Dept. of Computer Science & Engineering, Korea University, Seoul, Korea
{jtlee,sbkim,song,rim}@nlp.korea.ac.kr

Abstract

Lexical gaps between queries and questions (documents) have been a major issue in question retrieval on large online question and answer (Q&A) collections. Previous studies address the issue by *implicitly* expanding queries with the help of translation models pre-constructed using statistical techniques. However, since it is possible for *unimportant* words (e.g., non-topical words, common words) to be included in the translation models, a lack of noise control on the models can cause degradation of retrieval performance. This paper investigates a number of empirical methods for eliminating unimportant words in order to construct *compact translation models* for retrieval purposes. Experiments conducted on a real world Q&A collection show that substantial improvements in retrieval performance can be achieved by using compact translation models.

1 Introduction

Community-driven question answering services, such as Yahoo! Answers[1] and Live Search QnA[2], have been rapidly gaining popularity among Web users interested in sharing information online. By inducing users to collaboratively submit questions and answer questions posed by other users, large amounts of information have been collected in the form of question and answer (Q&A) pairs in recent years. This user-generated information is a valuable resource for many information seekers, because users can acquire information straightforwardly by searching through answered questions that satisfy their information need.

[1]http://answers.yahoo.com/
[2]http://qna.live.com/

Retrieval models for such Q&A collections should manage to handle the lexical gaps or word mismatches between user questions (queries) and answered questions in the collection. Consider the two following examples of questions that are semantically similar to each other:

- *"Where can I get cheap airplane tickets?"*
- *"Any travel website for low airfares?"*

Conventional word-based retrieval models would fail to capture the similarity between the two, because they have no words in common. To bridge the query-question gap, prior work on Q&A retrieval by Jeon et al. (2005) implicitly expands queries with the use of pre-constructed translation models, which lets you generate query words not in a question by translation to alternate words that are related. In practice, these translation models are often constructed using statistical machine translation techniques that primarily rely on word co-occurrence statistics obtained from parallel strings (e.g., question-answer pairs).

A critical issue of the translation-based approaches is the quality of translation models constructed in advance. If no noise control is conducted during the construction, it is possible for translation models to contain "unnecessary" translations (i.e., translating a word into an unimportant word, such as a non-topical or common word). In the query expansion viewpoint, an attempt to identify and decrease the proportion of unnecessary translations in a translation model may produce an effect of "selective" implicit query expansion and result in improved retrieval. However, prior work on translation-based Q&A retrieval does not recognize this issue and uses the translation model as it is; essentially no attention seems to have been paid to improving the performance of the translation-based approach by enhancing the quality of translation models.

In this paper, we explore a number of empirical methods for selecting and eliminating unimportant words from parallel strings to avoid unnecessary translations from being learned in translation models built for retrieval purposes. We use the term *compact translation models* to refer to the resulting models, since the total number of parameters for modeling translations would be minimized naturally. We also present experiments in which compact translation models are used in Q&A retrieval. The main goal of our study is to investigate if and how compact translation models can improve the performance of Q&A retrieval.

The rest of this paper is organized as follows. The next section introduces a translation-based retrieval model and accompanying techniques used to retrieve query-relevant questions. Section 3 presents a number of empirical ways to select and eliminate unimportant words from parallel strings for training compact translation models. Section 4 summarizes the compact translation models we built for retrieval experiments. Section 5 presents and discusses the results of retrieval experiments. Section 6 presents related works. Finally, the last section concludes the paper and discusses future directions.

2 Translation-based Retrieval Model

This section introduces the translation-based language modeling approach to retrieval that has been used to bridge the lexical gap between queries and already-answered questions in this paper.

In the basic language modeling framework for retrieval (Ponte and Croft, 1998), the similarity between a query Q and a document D for ranking may be modeled as the probability of the document language model M_D built from D generating Q:

$$sim(Q, D) \approx P(Q|M_D) \tag{1}$$

Assuming that query words occur independently given a particular document language model, the query-likelihood $P(Q|M_D)$ is calculated as:

$$P(Q|M_D) = \prod_{q \in Q} P(q|M_D) \tag{2}$$

where q represents a query word.

To avoid zero probabilities in document language models, a mixture between a document-specific multinomial distribution and a multinomial distribution estimated from the entire document collection is widely used in practice:

$$\begin{aligned} P(Q|M_D) \;\; = \;\; & \prod_{q \in Q} \Big[(1-\lambda) \cdot P(q|M_D) \\ & + \lambda \cdot P(q|M_C)\Big] \end{aligned} \tag{3}$$

where $0 < \lambda < 1$ and M_C represents a language model built from the entire collection. The probabilities $P(w|M_D)$ and $P(w|M_C)$ are calculated using maximum likelihood estimation.

The basic language modeling framework does not address the issue of lexical gaps between queries and question. Berger and Lafferty (1999) viewed information retrieval as statistical document-query translation and introduced translation models to map query words to document words. Assuming that a translation model can be represented by a conditional probability distribution of translation $T(\cdot|\cdot)$ between words, we can model $P(q|M_D)$ in Equation 3 as:

$$P(q|M_D) = \sum_{w \in D} T(q|w) P(w|M_D) \tag{4}$$

where w represents a document word.[3]

The translation probability $T(q|w)$ virtually represents the degree of relationship between query word q and document word w captured in a different, machine translation setting. Then, in the traditional information retrieval viewpoint, the use of translation models produce an implicit query expansion effect, since query words not in a document are mapped to related words in the document. This implies that translation-based retrieval models would make positive contributions to retrieval performance only when the pre-constructed translation models have reliable translation probability distributions.

[3]The formulation of our retrieval model is basically equivalent to the approach of Jeon et al. (2005).

2.1 IBM Translation Model 1

Obviously, we need to build a translation model in advance. Usually the IBM Model 1, developed in the statistical machine translation field (Brown et al., 1993), is used to construct translation models for retrieval purposes in practice. Specifically, given a number of parallel strings, the IBM Model 1 learns the translation probability from a source word s to a target word t as:

$$T(t|s) = \lambda_s^{-1} \sum_i^N c(t|s; J_i) \qquad (5)$$

where λ_s is a normalization factor to make the sum of translation probabilities for the word s equal to 1, N is the number of parallel string pairs, and J_i is the ith parallel string pair. $c(t|s; J_i)$ is calculated as:

$$\begin{aligned} c(t|s; J_i) &= \left(\frac{P(t|s)}{P(t|s_1) + \cdots + P(t|s_n)}\right) \\ &\quad \times freq_{t,J_i} \times freq_{s,J_i} \qquad (6) \end{aligned}$$

where $\{s_1, \ldots, s_n\}$ are words in the source text in J^i. $freq_{t,J^i}$ and $freq_{s,J^i}$ are the number of times that t and s occur in J_i, respectively.

Given the initial values of $T(t|s)$, Equations (5) and (6) are used to update $T(t|s)$ repeatedly until the probabilities converge, in an EM-based manner.

Note that the IBM Model 1 solely relies on word co-occurrence statistics obtained from parallel strings in order to learn translation probabilities. This implies that if parallel strings have unimportant words, a resulted translation model based on IBM Model 1 may contain unimportant words with non-zero translation probabilities.

We alleviate this drawback by eliminating unimportant words from parallel strings, avoiding them from being included in the conditional translation probability distribution. This naturally induces the construction of compact translation models.

2.2 Gathering Parallel Strings from Q&A Collections

The construction of statistical translation models previously discussed requires a corpus consisting of parallel strings. Since monolingual parallel texts are generally not available in real world, one must artificially generate a "synthetic" parallel corpus.

Question and answer as parallel pairs: The simplest approach is to directly employ questions and their answers in the collections by setting either as source strings and the other as target strings, with the assumption that a question and its corresponding answer are naturally parallel to each other. Formally, if we have a Q&A collection as $C = \{D_1, D_2, \ldots, D_n\}$, where D_i refers to an ith Q&A data consisting of a question q_i and its answer a_i, we can construct a parallel corpus C' as $\{(q_1, a_1), \ldots, (q_n, a_n)\} \cup \{(a_1, q_1), \ldots, (a_n, q_n)\} = C'$ where each element (s, t) refers to a parallel pair consisting of source string s and target string t. The number of parallel string samples would eventually be twice the size of the collections.

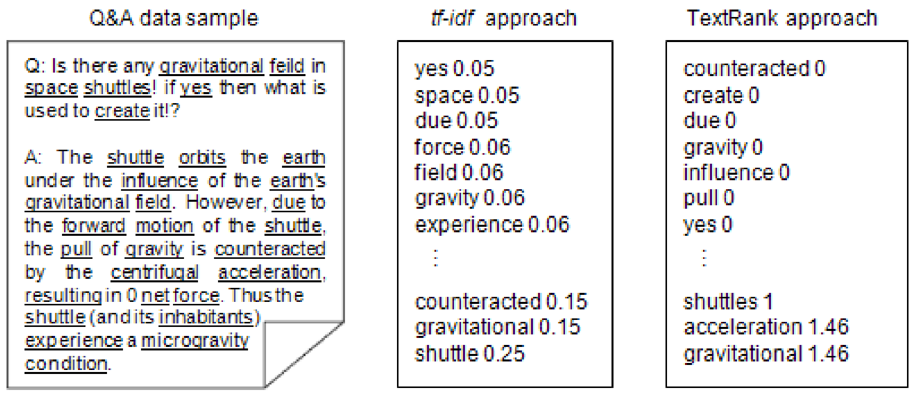
Similar questions as parallel pairs: Jeon et al. (2005) proposed an alternative way of automatically collecting a relatively larger set of parallel strings from Q&A collections. Motivated by the observation that many semantically identical questions can be found in typical Q&A collections, they used similarities between answers calculated by conventional word-based retrieval models to automatically group questions in a Q&A collection as pairs. Formally, two question strings q_i and q_j would be included in a parallel corpus C' as $\{(q_i, q_j), (q_j, q_i)\} \subset C'$ only if their answer strings a_i and a_j have a similarity higher than a pre-defined threshold value. The similarity is calculated as the reverse of the harmonic mean of ranks as $sim(a_i, a_j) = \frac{1}{2}(\frac{1}{r_j} + \frac{1}{r_i})$, where r_j and r_i refer to the rank of the a_j and a_i when a_i and a_j are given as queries, respectively. This approach may artificially produce much more parallel string pairs for training the IBM Model 1 than the former approach, depending on the threshold value.[4]

To our knowledge, there has not been any study comparing the effectiveness of the two approaches yet. In this paper, we try both approaches and compare the effectiveness in retrieval performance.

3 Eliminating Unimportant Words

We adopt a term weight ranking approach to identify and eliminate unimportant words from parallel strings, assuming that a word in a string is unim-

[4]We have empirically set the threshold (0.05) for our experiments.


Q&A data sample

Q: Is there any gravitational feild in space shuttles! if yes then what is used to create it!?

A: The shuttle orbits the earth under the influence of the earth's gravitational field. However, due to the forward motion of the shuttle, the pull of gravity is counteracted by the centrifugal acceleration, resulting in 0 net force. Thus the shuttle (and its inhabitants) experience a microgravity condition.

tf-idf approach

yes	0.05
space	0.05
due	0.05
force	0.06
field	0.06
gravity	0.06
experience	0.06
⋮	
counteracted	0.15
gravitational	0.15
shuttle	0.25

TextRank approach

counteracted	0
create	0
due	0
gravity	0
influence	0
pull	0
yes	0
⋮	
shuttles	1
acceleration	1.46
gravitational	1.46

Figure 1: Term weighting results of tf-idf and TextRank (window=3). Weighting is done on underlined words only.

portant if it holds a relatively low significance in the document (Q&A pair) of which the string is originally taken from. Some issues may arise:

- How to assign a weight to each word in a document for term ranking?
- How much to remove as unimportant words from the ranked list?

The following subsections discuss strategies we use to handle each of the issues above.

3.1 Assigning Term Weights

In this section, the two different term weighting strategies are introduced.

tf-idf: The use of *tf-idf* weighting on evaluating how unimportant a word is to a document seems to be a good idea to begin with. We have used the following formulas to calculate the weight of word w in document D:

$$tf\text{-}idf_{w,D} = tf_{w,D} \times idf_w \tag{7}$$

$$tf_{w,D} = \frac{freq_{w,D}}{|D|}, \quad idf_w = \log \frac{|C|}{df_w}$$

where $freq_{w,D}$ refers to the number of times w occurs in D, $|D|$ refers to the size of D (in words), $|C|$ refers to the size of the document collection, and df_w refers to the number of documents where w appears. Eventually, words with low *tf-idf* weights may be considered as unimportant.

TextRank: The task of term weighting, in fact, has been often applied to the keyword extraction task in natural language processing studies. As an alternative term weighting approach, we have used a variant of Mihalcea and Tarau (2004)'s TextRank, a graph-based ranking model for keyword extraction which achieves state-of-the-art accuracy without the need of deep linguistic knowledge or domain-specific corpora.

Specifically, the ranking algorithm proceeds as follows. First, words in a given document are added as vertices in a graph G. Then, edges are added between words (vertices) if the words co-occur in a fixed-sized window. The number of co-occurrences becomes the weight of an edge. When the graph is constructed, the score of each vertex is initialized as 1, and the PageRank-based ranking algorithm is run on the graph iteratively until convergence. The TextRank score of a word w in document D at kth iteration is defined as follows:

$$R^k_{w,D} = (1-d) + d \cdot \sum_{\forall j:(i,j)\in G} \frac{e_{i,j}}{\sum_{\forall l:(j,l)\in G} e_{j,l}} R^{k-1}_{w,D} \tag{8}$$

where d is a damping factor usually set to 0.85, and $e_{i,j}$ is an edge weight between i and j.

The assumption behind the use of the variant of TextRank is that a word is likely to be an important word in a document if it co-occurs frequently with other important words in the document. Eventually, words with low TextRank scores may be considered as unimportant. The main differences of TextRank compared to *tf-idf* is that it utilizes the context information of words to assign term weights.

Figure 1 demonstrates that term weighting results of TextRank and *tf-idf* are greatly different. Notice that TextRank assigns low scores to words that co-

Corpus: (Q‖A)	Vocabulary Size (%chg)		Average Translations (%chg)	
	tf-idf	*TextRank*	*tf-idf*	*TextRank*
Initial	90,441		73	
25%Removal	90,326 (∇0.1%)	73,021 (∇19.3%)	73 (∇0.0%)	44 (∇39.7%)
50%Removal	90,230 (∇0.2%)	72,225 (∇20.1%)	72 (∇1.4%)	43 (∇41.1%)
75%Removal	88,763 (∇1.9%)	65,268 (∇27.8%)	53 (∇27.4%)	38 (∇47.9%)
Avg.Score	66,412 (∇26.6%)	31,849 (∇64.8%)	14 (∇80.8%)	18 (∇75.3%)

Table 1: Impact of various word elimination strategies on translation model construction using (Q‖A) corpus.

Corpus: (Q‖Q)	Vocabulary Size (%chg)		Average Translations (%chg)	
	tf-idf	*TextRank*	*tf-idf*	*TextRank*
Initial	34,485		442	
25%Removal	34,374 (∇0.3%)	26,900 (∇22.0%)	437 (∇1.1%)	282 (∇36.2%)
50%Removal	34,262 (∇0.6%)	26,421 (∇23.4%)	423 (∇4.3%)	274 (∇38.0%)
75%Removal	32,813 (∇4.8%)	23,354 (∇32.3%)	288 (∇34.8%)	213 (∇51.8%)
Avg.Score	28,613 (∇17.0%)	16,492 (∇52.2%)	163 (∇63.1%)	164 (∇62.9%)

Table 2: Impact of various word elimination strategies on translation model construction using (Q‖Q) corpus.

occur only with stopwords. This implies that TextRank weighs terms more "strictly" than the *tf-idf* approach, with use of contexts of words.

3.2 Deciding the Quantity to be Removed from Ranked List

Once a final score (either *tf-idf* or TextRank score) is obtained for each word, we create a list of words ranked in decreasing order of their scores and eliminate the ones at lower ranks as unimportant words. The question here is how to decide the proportion or quantity to be removed from the ranked list.

Removing a fixed proportion: The first approach we have used is to decide the number of unimportant words based on the size of the original string. For our experiments, we manually vary the proportion to be removed as 25%, 50%, and 75%. For instance, if the proportion is set to 50% and an original string consists of ten words, at most five words would be remained as important words.

Using average score as threshold: We also have used an alternate approach to deciding the quantity. Instead of eliminating a *fixed* proportion, words are removed if their score is lower than the average score of all words in a document. This approach decides the proportion to be removed more flexibly than the former approach.

4 Building Compact Translation Models

We have initially built two parallel corpora from a Q&A collection[5], denoted as (Q‖A) corpus and (Q‖Q) corpus henceforth, by varying the methods in which parallel strings are gathered (described in Section 2.2). The (Q‖A) corpus consists of 85,938 parallel string pairs, and the (Q‖Q) corpus contains 575,649 parallel string pairs.

In order to build compact translation models, we have preprocessed the parallel corpus using different word elimination strategies so that unimportant words would be removed from parallel strings. We have also used a stoplist[6] consisting of 429 words to remove stopwords. The out-of-the-box GIZA++[7] (Och and Ney, 2004) has been used to learn translation models using the pre-processed parallel corpus for our retrieval experiments. We have also trained initial translation models, using a parallel corpus from which only the stopwords are removed, to compare with the compact translation models.

Eventually, the number of parameters needed for modeling translations would be minimized if unimportant words are eliminated with different ap-

[5]Details on this data will be introduced in the next section.
[6]http://truereader.com/manuals/onix/stopwords1.html
[7]http://www.fjoch.com/GIZA++.html

proaches. Table 1 and 2 shows the impact of various word elimination strategies on the construction of compact translation models using the (Q||A) corpus and the (Q||Q) corpus, respectively. The two tables report the size of the vocabulary contained and the average number of translations per word in the resulting compact translation models, along with percentage decreases with respect to the initial translation models in which only stopwords are removed. We make these observations:

- The translation models learned from the (Q||Q) corpus have *less* vocabularies but *more* average translations per word than the ones learned from the (Q||A) corpus. This result implies that a large amount of noise may have been created inevitably when a large number of parallel strings (pairs of similar questions) were artificially gathered from the Q&A collection.
- The TextRank strategy tends to eliminate larger sets of words as unimportant words than the *tf-idf* strategy when a fixed proportion is removed, regardless of the corpus type. Recall that the TextRank approach assigns weights to words more strictly by using contexts of words.
- The approach to remove words according to the average weight of a document (denoted as *Avg.Score*) tends to eliminate relatively larger portions of words as unimportant words than any of the fixed-proportion strategies, regardless of either the corpus type or the ranking strategy.

5 Retrieval Experiments

Experiments have been conducted on a real world Q&A collection to demonstrate the effectiveness of compact translation models on Q&A retrieval.

5.1 Experimental Settings

In this section, four experimental settings for the Q&A retrieval experiments are described in detail.

Data: For the experiments, Q&A data have been collected from the *Science* domain of Yahoo! Answers, one of the most popular community-based question answering service on the Web. We have obtained a total of 43,001 questions with a best answer (selected either by the questioner or by votes of other users) by recursively traversing subcategories of the *Science* domain, with up to 1,000 question pages retrieved.[8]

Among the obtained Q&A pairs, 32 Q&A pairs have been randomly selected as the test set, and the remaining 42,969 questions have been the reference set to be retrieved. Each Q&A pair has three text fields: question title, question content, and answer.[9] The fields of each Q&A pair in the test set are considered as various test queries; the question title, the question content, and the answer are regarded as a short query, a long query, and a supplementary query, respectively. We have used long queries and supplementary queries only in the relevance judgment procedure. All retrieval experiments have been conducted using short queries only.

Relevance judgments: To find relevant Q&A pairs given a short query, we have employed a pooling technique used in the TREC conference series. We have pooled the top 40 Q&A pairs from each retrieval results generated by varying the retrieval algorithms, the search field, and the query type. Popular word-based models, including the Okapi BM25, query-likelihood language model, and previous translation-based models (Jeon et al., 2005), have been used.[10]

Relevance judgments have been done by two student volunteers (both fluent in English). Since many community-based question answering services present their search results in a hierarchical fashion (*i.e.* a list of relevant questions is shown first, and then the user chooses a specific question from the list to see its answers), a Q&A pair has been judged as relevant if its question is semantically similar to the query; neither quality nor rightness of the answer has not been considered. When a disagreement has been made between two volunteers, one of the authors has made the final judgment. As a result, 177 relevant Q&A pairs have been found in total for the 32 short queries.

Baseline retrieval models: The proposed ap-

[8]Yahoo! Answers did not expose additional question pages to external requests at the time of collecting the data.

[9]When collecting parallel strings from the Q&A collection, we have put together the question title and the question content as one question string.

[10]The retrieval model using compact translation models has not been used in the pooling procedure.

proach to Q&A retrieval using compact translation models (denoted as CTLM henceforth) is compared to three baselines:

QLM: Query-likelihood language model for retrieval (equivalent to Equation 3, without use of translation models). This model represents word-based retrieval models widely used in practice.

TLM(Q||Q): Translation-based language model for question retrieval (Jeon et al., 2005). This model uses IBM Model 1 learned from the (Q||Q) corpus of which stopwords are removed.

TLM(Q||A): A variant of the translation-based approach. This model uses IBM model 1 learned from the (Q||A) corpus.

Evaluation metrics: We have reported the retrieval performance in terms of Mean Average Precision (MAP) and Mean R-Precision (R-Prec).

Average Precision can be computed based on the precision at each relevant document in the ranking. Mean Average Precision is defined as the mean of the Average Precision values across the set of all queries:

$$MAP(Q) = \frac{1}{|Q|}\sum_{q \in Q}\frac{1}{m_q}\sum_{k=1}^{m_q} Precision(R_k) \quad (9)$$

where Q is the set of test queries, m_q is the number of relevant documents for a query q, R_k is the set of ranked retrieval results from the top until rank position k, and $Precision(R_k)$ is the fraction of relevant documents in R_k (Manning et al., 2008).

R-Precision is defined as the precision after R documents have been retrieved where R is the number of relevant documents for the current query (Buckley and Voorhees, 2000). Mean R-Precision is the mean of the R-Precisions across the set of all queries.

We take MAP as our primary evaluation metric.

5.2 Experimental Results

Preliminary retrieval experiments have been conducted using the baseline QLM and different fields of Q&A data as retrieval unit. Table 3 shows the effectiveness of each field.

The results imply that the question title field is the most important field in our Yahoo! Answers collection; this also supports the observation presented by

Retrieval unit	**MAP**	**R-Prec**
Question title	**0.1031**	**0.2396**
Question content	0.0422	0.0999
Answer	0.0566	0.1062

Table 3: Preliminary retrieval results.

Model	**MAP (%chg)**	**R-Prec (%chg)**
QLM	0.1031	0.2396
TLM(Q||Q)*	0.1121 (△9%)	0.2251 (∇6%)
CTLM(Q||Q)	**0.1415** (△37%)	**0.2425** (△1%)
TLM(Q||A)	0.1935 (△88%)	0.3135 (△31%)
CTLM(Q||A)	**0.2095** (△103%)	**0.3585** (△50%)

Table 4: Comparisons with three baseline retrieval models. * indicates that it is equivalent to Jeon et al. (2005)'s approach. MAP improvements of CTLMs have been tested to be statistically significant using paired t-test.

Jeon et al. (2005). Based on the preliminary observations, all retrieval models tested in this paper have ranked Q&A pairs according to the similarity scores between queries and question titles.

Table 4 presents the comparison results of three baseline retrieval models and the proposed CTLMs. For each method, the best performance after empirical λ parameter tuning according to MAP is presented.

Notice that both the TLMs and CTLMs have outperformed the word-based QLM. This implies that word-based models that do not address the issue of lexical gaps between queries and questions often fail to retrieve relevant Q&A data that have little word overlap with queries, as noted by Jeon et al. (2005).

Moreover, notice that the proposed CTLMs have achieved significantly better performances in all evaluation metrics than both QLM and TLMs, regardless of the parallel corpus in which the incorporated translation models are trained from. This is a clear indication that the use of compact translation models built with appropriate word elimination strategies is effective in closing the query-question lexical gaps

(Q‖Q)	MAP (%chg)	
	tf-idf	*TextRank*
Initial	0.1121	
25%Rmv	0.1141 (△1.8)	0.1308 (△16.7)
50%Rmv	0.1261 (△12.5)	0.1334 (△19.00)
75%Rmv	0.1115 (▽0.5)	0.1160 (△3.5)
Avg.Score	0.1056 (▽5.8)	**0.1415 (△26.2)**

Table 5: Contributions of various word elimination strategies on MAP performance of `CTLM(Q‖Q)`.

(Q‖A)	MAP (%chg)	
	tf-idf	*TextRank*
Initial	0.1935	
25%Rmv	**0.2095 (△8.3)**	0.1733 (▽10.4)
50%Rmv	0.2085 (△7.8)	0.1623 (▽16.1)
75%Rmv	0.1449 (▽25.1)	0.1515 (▽21.7)
Avg.Score	0.1168 (▽39.6)	0.1124 (▽41.9)

Table 6: Contributions of various word elimination strategies on MAP performance of `CTLM(Q‖A)`.

for improving the performance of question retrieval in the context of language modeling framework.

Note that the retrieval performance varies by the type of training corpus; `CTLM(Q‖A)` has outperformed `CTLM(Q‖Q)` significantly. This proves the statement we made earlier that the (Q‖Q) corpus would contain much noise since the translation models learned from the (Q‖Q) corpus tend to have smaller vocabulary sizes but significantly more average translations per word than the ones learned from the (Q‖A) corpus.

Table 5 and 6 show the effect of various word elimination strategies on the retrieval performance of `CTLM`s in which the incorporated compact translation models are trained from the (Q‖Q) corpus and the (Q‖A) corpus, respectively. It is interesting to note that the importance of modifications in word elimination strategies also varies by the type of training corpus.

The retrieval results indicate that when the translation model is trained from the “less noisy” (Q‖A) corpus, eliminating a relatively large proportions of words may hurt the retrieval performance of `CTLM`. In the case when the translation model is trained from the “noisy” (Q‖Q) corpus, a better retrieval performance may be achieved if words are eliminated appropriately to a certain extent.

In terms of weighting scheme, the TextRank approach, which is more “strict” than *tf-idf* in eliminating unimportant words, has led comparatively higher retrieval performances on all levels of removal quantity when the translation model has been trained from the “noisy” (Q‖Q) corpus. On the contrary, the “less strict” *tf-idf* approach has led better performances when the translation model has been trained from the “less noisy” (Q‖A) corpus.

In summary, the results imply that the performance of translation-based retrieval models can be significantly improved when strategies for building of compact translation models are chosen properly, regarding the expected noise level of the parallel corpus for training the translation models. In a case where a noisy parallel corpus is given for training of translation models, it is better to get rid of noise as much as possible by using “strict” term weighting algorithms; when a less noisy parallel corpus is given for building the translation models, a tolerant approach would yield better retrieval performance.

6 Related Works

Our work is most closely related to Jeon et al. (2005)’s work, which addresses the issue of word mismatch between queries and questions in large online Q&A collections by using translation-based methods. Apart from their work, there have been some related works on applying translation-based methods for retrieving FAQ data. Berger et al. (2000) report some of the earliest work on FAQ retrieval using statistical retrieval models, including translation-based approaches, with a small set of FAQ data. Soricut and Brill (2004) present an answer passage retrieval system that is trained from 1 million FAQs collected from the Web using translation methods. Riezler et al. (2007) demonstrate the advantages of translation-based approach to answer retrieval by utilizing a more complex translation model also trained from a large amount of data extracted from FAQs on the Web. Although all of these translation-based approaches are based on the statistical translation models, including the IBM Model 1, none of them focus on addressing the noise issues in translation models.

7 Conclusion and Future Work

Bridging the query-question gap has been a major issue in retrieval models for large online Q&A collections. In this paper, we have shown that the performance of translation-based retrieval on real online Q&A collections can be significantly improved by using compact translation models of which the noise (unimportant word translations) is properly reduced. We have also observed that the performance enhancement may be achieved by choosing the appropriate strategies regarding the strictness of various term weighting algorithms and the expected noise level of the parallel data for learning such translation models.

Future work will focus on testing the effectiveness of the proposed method on a larger set of Q&A collections with broader domains. Since the proposed approach cannot handle many-to-one or one-to-many word transformations, we also plan to investigate the effectiveness of phrase-based translation models in closing gaps between queries and questions for further enhancement of Q&A retrieval.

Acknowledgments

This work was supported by Microsoft Research Asia. Any opinions, findings, and conclusions or recommendations expressed above are those of the authors and do not necessarily reflect the views of the sponsor.

References

Adam Berger, Rich Caruana, David Cohn, Dayne Freitag, and Vibhu Mittal. 2000. Bridging the Lexical Chasm: Statistical Approaches to Answer-Finding. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 192–199.

Adam Berger and John Lafferty. 1999. Information Retrieval as Statistical Translation. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 222–229.

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.

Chris Buckley and Ellen M. Voorhees. 2000. Evaluating Evaluation Measure Stability. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 33–40.

Jiwoon Jeon, W. Bruce Croft, and Joon Ho Lee. 2005. Finding Similar Questions in Large Question and Answer Archives. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 84–90.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411.

Franz J. Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51.

Jay M. Ponte and W. Bruce Croft. 1998. A Language Modeling Approach to Information Retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281.

Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. 2007. Statistical Machine Translation for Query Expansion in Answer Retrieval. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 464–471.

Radu Soricut and Eric Brill. 2004. Automatic Question Answering: Beyond the Factoid. In *Proceedings of the 2004 Human Language Technology and Conference of the North American Chapter of the Association for Computational Linguistics*, pages 57–64.

Scalable Language Processing Algorithms for the Masses: A Case Study in Computing Word Co-occurrence Matrices with MapReduce

Jimmy Lin
The iSchool, University of Maryland
National Center for Biotechnology Information, National Library of Medicine
jimmylin@umd.edu

Abstract

This paper explores the challenge of scaling up language processing algorithms to increasingly large datasets. While cluster computing has been available in commercial environments for several years, academic researchers have fallen behind in their ability to work on large datasets. I discuss two barriers contributing to this problem: lack of a suitable programming model for managing concurrency and difficulty in obtaining access to hardware. Hadoop, an open-source implementation of Google's MapReduce framework, provides a compelling solution to both issues. Its simple programming model hides system-level details from the developer, and its ability to run on commodity hardware puts cluster computing within the reach of many academic research groups. This paper illustrates these points with a case study in building word co-occurrence matrices from large corpora. I conclude with an analysis of an alternative computing model based on renting instead of buying computer clusters.

1 Introduction

Over the past couple of decades, the field of computational linguistics (and more broadly, human language technologies) has seen the emergence and later dominance of empirical techniques and data-driven research. Concomitant with this trend is a coherent research thread that focuses on exploiting increasingly-large datasets. Banko and Brill (2001) were among the first to demonstrate the importance of dataset size as a significant factor governing prediction accuracy in a supervised machine learning task. In fact, they argued that size of training set was perhaps more important than the choice of machine learning algorithm itself. Similarly, experiments in question answering have shown the effectiveness of simple pattern-matching techniques when applied to large quantities of data (Brill et al., 2001; Dumais et al., 2002). More recently, this line of argumentation has been echoed in experiments with Web-scale language models. Brants et al. (2007) showed that for statistical machine translation, a simple smoothing technique (dubbed *Stupid Backoff*) approaches the quality of the Kneser-Ney algorithm as the amount of training data increases, and with the simple method one can process significantly more data.

Challenges in scaling algorithms to increasingly-large datasets have become a serious issue for researchers. It is clear that datasets readily available today and the types of analyses that researchers wish to conduct have outgrown the capabilities of individual computers. The only practical recourse is to distribute the computation across multiple cores, processors, or machines. The consequences of failing to scale include misleading generalizations on artificially small datasets and limited practical applicability in real-world contexts, both undesirable.

This paper focuses on two barriers to developing scalable language processing algorithms: challenges associated with parallel programming and access to hardware. Google's MapReduce framework (Dean and Ghemawat, 2004) provides an attractive programming model for developing scalable algorithms, and with the release of Hadoop, an open-source implementation of MapReduce lead by Yahoo, cost-effective cluster computing is within the reach of most academic research groups. It is emphasized that this work focuses on large-data algorithms from the perspective of academia—colleagues in commercial environments have long enjoyed the advantages of cluster computing. However, it is only recently that such capabilities have become *practical* for academic research groups. These points are illustrated by a case study in building large word co-occurrence matrices, a simple task that underlies many NLP algorithms.

The remainder of the paper is organized as follows: the next section overviews the MapReduce framework and why it provides a compelling solution to the issues sketched above. Section 3 introduces the task of building word co-occurrence matrices, which provides an illustrative case study. Two separate algorithms are presented in Section 4. The experimental setup is described in Section 5, followed by presentation of results in Section 6. Implications and generalizations are discussed following that. Before concluding, I explore an alternative model of computing based on renting instead of buying hardware, which makes cluster computing practical for *everyone*.

2 MapReduce

The only practical solution to large-data challenges today is to distribute the computation across multiple cores, processors, or machines. The development of parallel algorithms involves a number of tradeoffs. First is that of cost: a decision must be made between "exotic" hardware (e.g., large shared memory machines, InfiniBand interconnect) and commodity hardware. There is significant evidence (Barroso et al., 2003) that solutions based on the latter are more cost effective—and for resource-constrained academic NLP groups, commodity hardware is often the only practical route.

Given appropriate hardware, researchers must still contend with the challenge of developing software. Quite simply, parallel programming is difficult. Due to communication and synchronization issues, concurrent operations are notoriously challenging to reason about. Reliability and fault tolerance become important design considerations on clusters containing large numbers of unreliable commodity parts. With traditional parallel programming models (e.g., MPI), the developer shoulders the burden of explicitly managing concurrency. As a result, a significant amount of the programmer's attention is devoted to system-level details, leaving less time for focusing on the actual problem.

Recently, MapReduce (Dean and Ghemawat, 2004) has emerged as an attractive alternative to existing parallel programming models. The MapReduce abstraction shields the programmer from having to explicitly worry about system-level issues such as synchronization, inter-process communication, and fault tolerance. The runtime is able to transparently distribute computations across large clusters of commodity hardware with good scaling characteristics. This frees the programmer to focus on solving the problem at hand.

MapReduce builds on the observation that many information processing tasks have the same basic structure: a computation is applied over a large number of records (e.g., Web pages, bitext pairs, or nodes in a graph) to generate partial results, which are then aggregated in some fashion. Naturally, the per-record computation and aggregation function vary according to task, but the basic structure remains fixed. Taking inspiration from higher-order functions in functional programming, MapReduce provides an abstraction at the point of these two operations. Specifically, the programmer defines a "mapper" and a "reducer" with the following signatures:

map: $(k_1, v_1) \rightarrow [(k_2, v_2)]$
reduce: $(k_2, [v_2]) \rightarrow [(k_3, v_3)]$

Key-value pairs form the basic data structure in MapReduce. The mapper is applied to every input key-value pair to generate an arbitrary number of intermediate key-value pairs ([...] is used to denote a list). The reducer is applied to all values associated with the same intermediate key to generate output key-value pairs. This two-stage processing structure is illustrated in Figure 1.

Under the framework, a programmer needs only to provide implementations of the mapper and reducer. On top of a distributed file system (Ghemawat et al., 2003), the runtime transparently handles all other aspects of execution, on clusters ranging from a few to a few thousand nodes. The runtime is responsible for scheduling map and reduce

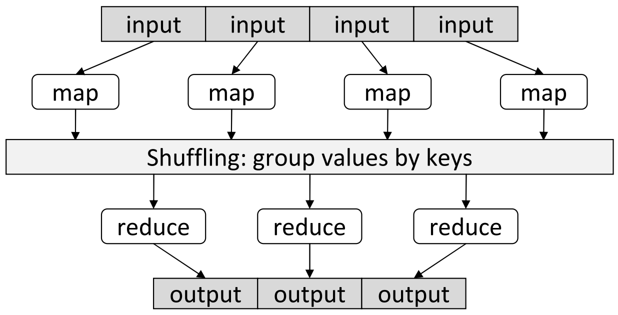


Figure 1: Illustration of the MapReduce framework: the "mapper" is applied to all input records, which generates results that are aggregated by the "reducer". The runtime groups together values by keys.

workers on commodity hardware assumed to be unreliable, and thus is tolerant to various faults through a number of error recovery mechanisms. In the distributed file system, data blocks are stored on the local disks of machines in the cluster—the MapReduce runtime handles the scheduling of mappers on machines where the necessary data resides. It also manages the potentially very large sorting problem between the map and reduce phases whereby intermediate key-value pairs must be grouped by key.

As an optimization, MapReduce supports the use of "combiners", which are similar to reducers except that they operate directly on the output of mappers (in memory, before intermediate output is written to disk). Combiners operate in isolation on each node in the cluster and cannot use partial results from other nodes. Since the output of mappers (i.e., the key-value pairs) must ultimately be shuffled to the appropriate reducer over a network, combiners allow a programmer to aggregate partial results, thus reducing network traffic. In cases where an operation is both associative and commutative, reducers can directly serve as combiners.

Google's proprietary implementation of MapReduce is in C++ and not available to the public. However, the existence of Hadoop, an open-source implementation in Java spearheaded by Yahoo, allows anyone to take advantage of MapReduce. The growing popularity of this technology has stimulated a flurry of recent work, on applications in machine learning (Chu et al., 2006), machine translation (Dyer et al., 2008), and document retrieval (Elsayed et al., 2008).

3 Word Co-occurrence Matrices

To illustrate the arguments outlined above, I present a case study using MapReduce to build word co-occurrence matrices from large corpora, a common task in natural language processing. Formally, the co-occurrence matrix of a corpus is a square $N \times N$ matrix where N corresponds to the number of unique words in the corpus. A cell m_{ij} contains the number of times word w_i co-occurs with word w_j within a specific context—a natural unit such as a sentence or a certain window of m words (where m is an application-dependent parameter). Note that the upper and lower triangles of the matrix are identical since co-occurrence is a symmetric relation.

This task is quite common in corpus linguistics and provides the starting point to many other algorithms, e.g., for computing statistics such as pointwise mutual information (Church and Hanks, 1990), for unsupervised sense clustering (Schütze, 1998), and more generally, a large body of work in lexical semantics based on distributional profiles, dating back to Firth (1957) and Harris (1968). The task also has applications in information retrieval, e.g., (Schütze and Pedersen, 1998; Xu and Croft, 1998), and other related fields as well. More generally, this problem relates to the task of estimating distributions of discrete events from a large number of observations (more on this in Section 7).

It is obvious that the space requirement for this problem is $O(N^2)$, where N is the size of the vocabulary, which for real-world English corpora can be hundreds of thousands of words. The computation of the word co-occurrence matrix is quite simple if the entire matrix fits into memory—however, in the case where the matrix is too big to fit in memory, a naive implementation can be very slow as memory is paged to disk. For large corpora, one needs to optimize disk access and avoid costly seeks. As illustrated in the next section, MapReduce handles exactly these issues transparently, allowing the programmer to express the algorithm in a straightforward manner.

A bit more discussion of the task before moving on: in many applications, researchers have discovered that building the complete word co-occurrence matrix may not be necessary. For example, Schütze (1998) discusses feature selection techniques in defining context vectors; Mohammad and Hirst (2006) present evidence that conceptual distance is better captured via distributional profiles mediated by thesaurus categories. These objections, however, miss the point—the focus of this paper is on practical cluster computing for academic researchers; this particular task serves merely as an illustrative example. In addition, for rapid prototyping, it may be useful to start with the complete co-occurrence matrix (especially if it can be built efficiently), and then explore how algorithms can be optimized for specific applications and tasks.

4 MapReduce Implementation

This section presents two MapReduce algorithms for building word co-occurrence matrices for large corpora. The goal is to illustrate how the problem can be concisely captured in the MapReduce programming model, and how the runtime hides many of the system-level details associated with distributed computing.

Pseudo-code for the first, more straightforward, algorithm is shown in Figure 2. Unique document ids and the corresponding texts make up the input key-value pairs. The mapper takes each input document and emits intermediate key-value pairs with each co-occurring word pair as the key and the integer one as the value. In the pseudo-code, EMIT denotes the creation of an intermediate key-value pair that is collected (and appropriately sorted) by the MapReduce runtime. The reducer simply sums up all the values associated with the same co-occurring word pair, arriving at the absolute counts of the joint event in the corpus (corresponding to each cell in the co-occurrence matrix).

For convenience, I refer to this algorithm as the "pairs" approach. Since co-occurrence is a symmetric relation, it suffices to compute half of the matrix. However, for conceptual clarity and to generalize to instances where the relation may not be symmetric, the algorithm computes the entire matrix.

The Java implementation of this algorithm is quite concise—less than fifty lines long. Notice the MapReduce runtime guarantees that all values associated with the same key will be gathered together at the reduce stage. Thus, the programmer does not need to explicitly manage the collection and distribution of partial results across a cluster. In addition, the programmer does not need to explicitly partition the input data and schedule workers. This example shows the extent to which distributed processing can be dominated by system issues, and how an appropriate abstraction can significantly simplify development.

```
1: procedure MAP_1($n, d$)
2:     for all $w \in d$ do
3:         for all $u \in$ NEIGHBORS($w$) do
4:             EMIT(($w, u$), 1)
1: procedure REDUCE_1($p, [v_1, v_2, \ldots]$)
2:     for all $v \in [v_1, v_2, \ldots]$ do
3:         $sum \leftarrow sum + v$
4:     EMIT($p, sum$)
```

Figure 2: Pseudo-code for the "pairs" approach for computing word co-occurrence matrices.

```
1: procedure MAP_2($n, d$)
2:     INITIALIZE($H$)
3:     for all $w \in d$ do
4:         for all $u \in$ NEIGHBORS($w$) do
5:             $H\{u\} \leftarrow H\{u\} + 1$
6:     EMIT($w, H$)
1: procedure REDUCE_2($w, [H_1, H_2, H_3, \ldots]$)
2:     INITIALIZE($H_f$)
3:     for all $H \in [H_1, H_2, H_3, \ldots]$ do
4:         MERGE($H_f, H$)
5:     EMIT($w, H_f$)
```

Figure 3: Pseudo-code for the "stripes" approach for computing word co-occurrence matrices.

It is immediately obvious that Algorithm 1 generates an immense number of key-value pairs. Although this can be mitigated with the use of a combiner (since addition is commutative and associative), the approach still results in a large amount of network traffic. An alternative approach is presented in Figure 3, first reported in Dyer et al. (2008). The major difference is that counts of co-occurring words are first stored in an associative array (H). The output of the mapper is a number of key-value pairs with words as keys and the corresponding associative arrays as the values. The reducer performs an element-wise sum of all associative arrays with the same key (denoted by the function MERGE), thus accumulating counts that correspond to the same cell in the co-occurrence matrix. Once again, a combiner can be used to cut down on the network traffic by merging partial results. In the final output, each key-value pair corresponds to a row in the word co-occurrence matrix. For convenience, I refer to this as the "stripes" approach.

Compared to the "pairs" approach, the "stripes" approach results in far fewer intermediate key-value pairs, although each is significantly larger (and there is overhead in serializing and deserializing associative arrays). A critical assumption of the "stripes" approach is that at any point in time, each associative array is small enough to fit into memory (otherwise, memory paging may result in a serious loss of efficiency). This is true for most corpora, since the size of the associative array is bounded by the vocabulary size. Section 6 compares the efficiency of both algorithms.[1]

5 Experimental Setup

Work reported in this paper used the English Gigaword corpus (version 3),[2] which consists of newswire documents from six separate sources, totaling 7.15 million documents (6.8 GB compressed, 19.4 GB uncompressed). Some experiments used only documents from the Associated Press Worldstream (APW), which contains 2.27 million documents (1.8 GB compressed, 5.7 GB uncompressed). By LDC's count, the entire collection contains approximately 2.97 billion words.

Prior to working with Hadoop, the corpus was first preprocessed. All XML markup was removed, followed by tokenization and stopword removal using standard tools from the Lucene search engine. All tokens were replaced with unique integers for a more efficient encoding. The data was then packed into a Hadoop-specific binary file format. The entire Gigaword corpus took up 4.69 GB in this format; the APW sub-corpus, 1.32 GB.

Initial experiments used Hadoop version 0.16.0 running on a 20-machine cluster (1 master, 19 slaves). This cluster was made available to the University of Maryland as part of the Google/IBM Academic Cloud Computing Initiative. Each machine has two single-core processors (running at either 2.4 GHz or 2.8 GHz), 4 GB memory. The cluster has an aggregate storage capacity of 1.7 TB. Hadoop ran on top of a virtualization layer, which has a small but measurable impact on performance; see (Barham et al., 2003). Section 6 reports experimental results using this cluster; Section 8 explores an alternative model of computing based on "renting cycles".

[1]Implementations of both algorithms are included in Cloud[9], an open source Hadoop library that I have been developing to support research and education, available from my homepage.

[2]LDC catalog number LDC2007T07

6 Results

First, I compared the running time of the "pairs" and "stripes" approaches discussed in Section 4. Running times on the 20-machine cluster are shown in Figure 4 for the APW section of the Gigaword corpus: the *x*-axis shows different percentages of the sub-corpus (arbitrarily selected) and the *y*-axis shows running time in seconds. For these experiments, the co-occurrence window was set to two, i.e., w_i is said to co-occur with w_j if they are no more than two words apart (after tokenization and stopword removal).

Results demonstrate that the stripes approach is far more efficient than the pairs approach: 666 seconds (11m 6s) compared to 3758 seconds (62m 38s) for the entire APW sub-corpus (improvement by a factor of 5.7). On the entire sub-corpus, the mappers in the pairs approach generated 2.6 billion intermediate key-value pairs totally 31.2 GB. After the combiners, this was reduced to 1.1 billion key-value pairs, which roughly quantifies the amount of data involved in the shuffling and sorting of the keys. On the other hand, the mappers in the stripes approach generated 653 million intermediate key-value pairs totally 48.1 GB; after the combiners, only 28.8 million key-value pairs were left. The stripes approach provides more opportunities for combiners to aggregate intermediate results, thus greatly reducing network traffic in the sort and shuffle phase.

Figure 4 also shows that both algorithms exhibit highly desirable scaling characteristics—linear in the corpus size. This is confirmed by a linear regression applied to the running time data, which yields R^2 values close to one. Given that the stripes algorithm is more efficient, it is used in the remainder of the experiments.

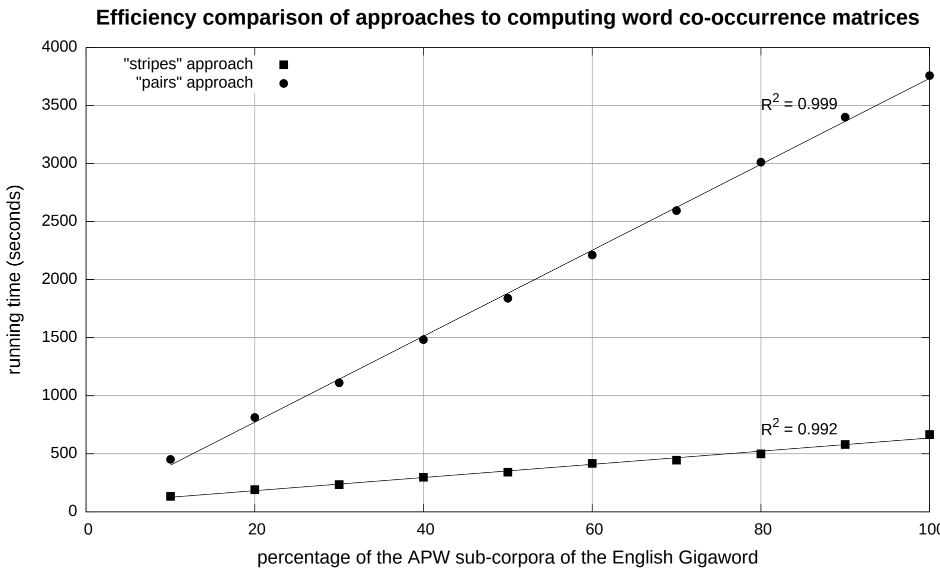


Figure 4: Running time of the two algorithms ("stripes" vs. "pairs") for computing word co-occurrence matrices on the APW section of the Gigaword corpus. The cluster used for this experiment contains 20 machines, each with two single-core processors.

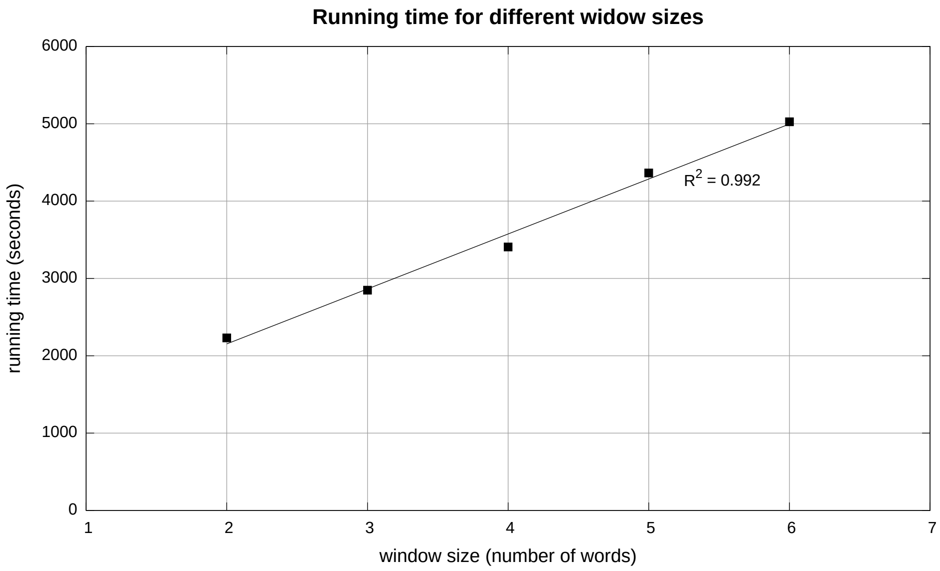


Figure 5: Running times for computing word co-occurrence matrices from the entire Gigaword corpus with varying window sizes. The cluster used for this experiment contains 20 machines, each with two single-core processors.

With a window size of two, computing the word co-occurrence matrix for the entire Gigaword corpus (7.15 million documents) takes 37m 11s on the 20-machine cluster. Figure 5 shows the running time as a function of window size. With a window of six words, running time on the complete Gigaword corpus rises to 1h 23m 45s. Once again, the stripes algorithm exhibits the highly desirable characteristic of linear scaling in terms of window size, as confirmed by the linear regression with an R^2 value very close to one.

7 Discussion

The elegance of the programming model and good scaling characteristics of resulting implementations make MapReduce a compelling tool for a variety of natural language processing tasks. In fact, MapReduce excels at a large class of problems in NLP that involves estimating probability distributions of discrete events from a large number of observations according to the maximum likelihood criterion:

$$P_{MLE}(B|A) = \frac{c(A,B)}{c(A)} = \frac{c(A,B)}{\sum_{B'} c(A,B')} \quad (1)$$

In practice, it matters little whether these events are words, syntactic categories, word alignment links, or any construct of interest to researchers. Absolute counts in the stripes algorithm presented in Section 4 can be easily converted into conditional probabilities by a final normalization step. Recently, Dyer et al. (2008) used this approach for word alignment and phrase extraction in statistical machine translation. Of course, many applications require smoothing of the estimated distributions—this problem also has known solutions in MapReduce (Brants et al., 2007).

Synchronization is perhaps the single largest bottleneck in distributed computing. In MapReduce, this is handled in the shuffling and sorting of key-value pairs between the map and reduce phases. Development of efficient MapReduce algorithms critically depends on careful control of intermediate output. Since the network link between different nodes in a cluster is by far the component with the largest latency, any reduction in the size of intermediate output or a reduction in the number of key-value pairs will have significant impact on efficiency.

8 Computing on Demand

The central theme of this paper is practical cluster computing for NLP researchers in the academic environment. I have identified two key aspects of what it means to be "practical": the first is an appropriate programming model for simplifying concurrency management; the second is access to hardware resources. The Hadoop implementation of MapReduce addresses the first point and to a large extent the second point as well. The cluster used for experiments in Section 6 is modest by today's standards and within the capabilities of many academic research groups. It is not even a requirement for the computers to be rack-mounted units in a machine room (although that is clearly preferable); there are plenty of descriptions on the Web about Hadoop clusters built from a handful of desktop machines connected by gigabit Ethernet.

Even without access to hardware, cluster computing remains within the reach of resource-constrained academics. "Utility computing" is an emerging concept whereby anyone can provision clusters on demand from a third-party provider. Instead of upfront capital investment to acquire a cluster and reoccurring maintenance and administration costs, one could "rent" computing cycles as they are needed—this is not a new idea (Rappa, 2004). One such service is provided by Amazon, called Elastic Compute Cloud (EC2).[3] With EC2, researchers could dynamically create a Hadoop cluster on-the-fly and tear down the cluster once experiments are complete. To demonstrate the use of this technology, I replicated some of the previous experiments on EC2 to provide a case study of this emerging model of computing.

Virtualized computation units in EC2 are called instances. At the time of these experiments, the basic instance offers, according to Amazon, 1.7 GB of memory, 1 EC2 Compute Unit (1 virtual core with 1 EC2 Compute Unit), and 160 GB of instance storage. Each instance-hour costs \$0.10 (all prices given in USD). Computational resources are simply charged by the instance-hour, so that a ten-instance cluster for ten hours costs the same as a hundred-instance cluster for one hour (both \$10)—the Amazon infrastructure allows one to dynamically provision and release resources as necessary. This is at-

[3]http://www.amazon.com/ec2

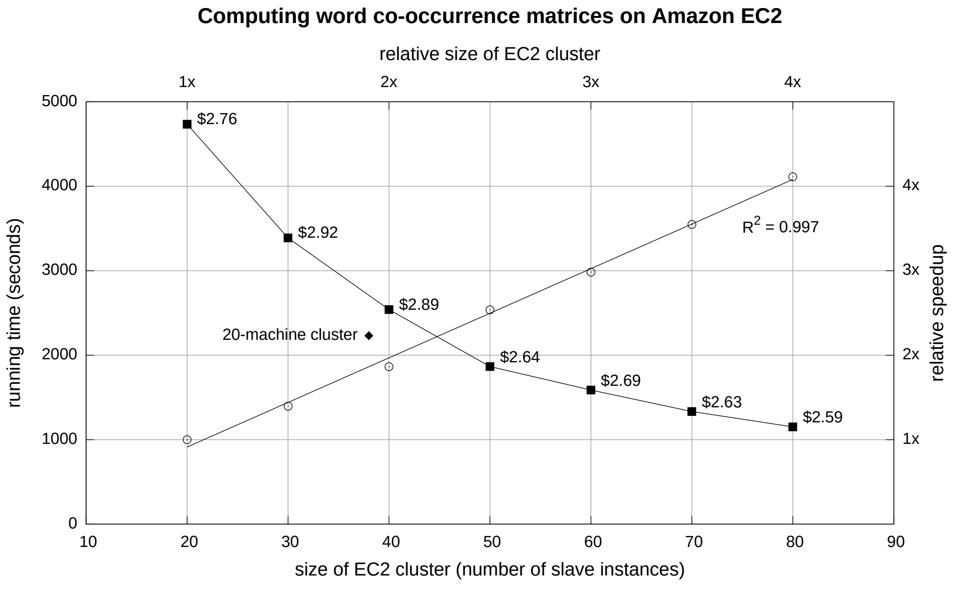


Figure 6: Running time analysis on Amazon EC2 with various cluster sizes; solid squares are annotated with the cost of each experiment. Alternate axes (circles) plot scaling characteristics in terms increasing cluster size.

tractive for researchers, who could on a limited basis allocate clusters much larger than they could otherwise afford if forced to purchase the hardware outright. Through virtualization technology, Amazon is able to parcel out allotments of processor cycles while maintaining high overall utilization across a data center and exploiting economies of scale.

Using EC2, I built word co-occurrence matrices from the entire English Gigaword corpus (window of two) on clusters of various sizes, ranging from 20 slave instances all the way up to 80 slave instances. The entire cluster consists of the slave instances plus a master controller instance that serves as the job submission queue; the clusters ran Hadoop version 0.17.0 (the latest release at the time these experiments were conducted). Running times are shown in Figure 6 (solid squares), with varying cluster sizes on the *x*-axis. Each data point is annotated with the cost of running the complete experiment.[4] Results show that computing the complete word co-occurrence matrix costs, quite literally, a couple of dollars—certainly affordable by any academic researcher without access to hardware. For reference, Figure 6 also plots the running time of the same experiment on the 20-machine cluster used in Section 6 (which contains 38 worker cores, each roughly comparable to an instance).

The alternate set of axes in Figure 6 shows the scaling characteristics of various cluster sizes. The circles plot the relative size and speedup of the EC2 experiments, with respect to the 20-slave cluster. The results show highly desirable linear scaling characteristics.

The above figures include only the cost of running the instances. One must additionally pay for bandwidth when transferring data in and out of EC2. At the time these experiments were conducted, Amazon charged \$0.10 per GB for data transferred in and \$0.17 per GB for data transferred out. To complement EC2, Amazon offers persistent storage via the Simple Storage Service (S3),[5] at a cost of \$0.15 per GB per month. There is no charge for data transfers between EC2 and S3. The availability of this service means that one can choose between paying for data transfer or paying for persistent storage on a cyclic basis—the tradeoff naturally depends on the amount of data and its permanence.

The cost analysis presented above assumes optimally-efficient use of Amazon's services; end-to-end cost might better quantify real-world usage conditions. In total, the experiments reported in this

[4]Note that Amazon bills in whole instance-hour increments; these figures assume fractional accounting.

[5]http://www.amazon.com/s3

section resulted in a bill of approximately thirty dollars. The figure includes all costs associated with instance usage and data transfer costs. It also includes time taken to learn the Amazon tools (I previously had no experience with either EC2 or S3) and to run preliminary experiments on smaller datasets (before scaling up to the complete corpus). The lack of fractional accounting on instance-hours contributed to the larger-than-expected costs, but such wastage would naturally be reduced with more experiments and higher sustained use. Overall, these cost appear to be very reasonable, considering that the largest cluster in these experiments (1 master + 80 slave instances) might be too expensive for most academic research groups to own and maintain.

Consider another example that illustrates the possibilities of utility computing. Brants et al. (2007) described experiments on building language models with increasingly-large corpora using MapReduce. Their paper reported experiments on a corpus containing 31 billion tokens (about an order of magnitude larger than the English Gigaword): on 400 machines, the model estimation took 8 hours.[6] With EC2, such an experiment would cost a few hundred dollars—sufficiently affordable that availability of data becomes the limiting factor, not computational resources themselves.

The availability of "computing-on-demand" services and Hadoop make cluster computing practical for academic researchers. Although Amazon is currently the most prominent provider of such services, they are not the sole player in an emerging market—in the future there will be a vibrant market with many competing providers. Considering the tradeoffs between "buying" and "renting", I would recommend the following model for an academic research group: purchase a modest cluster for development and for running smaller experiments; use a computing-on-demand service for scaling up and for running larger experiments (since it would be more difficult to economically justify a large cluster if it does not receive high sustained utilization).

If the concept of utility computing takes hold, it would have a significant impact on computer science research in general: the natural implication is that algorithms should not only be analyzed in traditional terms such as asymptotic complexity, but also in terms of monetary costs, in relationship to dataset and cluster size. One can argue that cost is a more direct and practical measure of algorithmic efficiency.

9 Conclusion

This paper address two challenges faced by academic research groups in scaling up natural language processing algorithms to large corpora: the lack of an appropriate programming model for expressing the problem and the difficulty in getting access to hardware. With this case study in building word co-occurrence matrices from large corpora, I demonstrate that MapReduce, via the open source Hadoop implementation, provides a compelling solution. A large class of algorithms in computational linguistics can be readily expressed in MapReduce, and the resulting code can be transparently distributed across commodity clusters. Finally, the "cycle-renting" model of computing makes access to large clusters affordable to researchers with limited resources. Together, these developments dramatically lower the entry barrier for academic researchers who wish to explore large-data issues.

Acknowledgments

This work was supported by the Intramural Research Program of the NIH, National Library of Medicine; NSF under awards IIS-0705832 and IIS-0836560; DARPA/IPTO Contract No. HR0011-06-2-0001 under the GALE program. Any opinions, findings, conclusions, or recommendations expressed in this paper are the author's and do not necessarily reflect those of the sponsors. I would like to thank Yahoo! for leading the development of Hadoop, IBM and Google for hardware support via the Academic Cloud Computing Initiative (ACCI), and Amazon for EC2/S3 support. This paper provides a neutral evaluation of EC2 and S3, and should not be interpreted as endorsement for the commercial services offered by Amazon. I wish to thank Philip Resnik and Doug Oard for comments on earlier drafts of this paper, and Ben Shneiderman for helpful editing suggestions. I am, as always, grateful to Esther and Kiri for their kind support.

[6]Brants et al. were affiliated with Google, so access to hardware was not an issue.

References

Michele Banko and Eric Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL 2001)*, pages 26–33, Toulouse, France.

Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. 2003. Xen and the art of virtualization. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP-03)*, pages 164–177, Bolton Landing, New York.

Luiz André Barroso, Jeffrey Dean, and Urs Hölzle. 2003. Web search for a planet: The Google cluster architecture. *IEEE Micro*, 23(2):22–28.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 858–867, Prague, Czech Republic.

Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Ng. 2001. Data-intensive question answering. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, pages 393–400, Gaithersburg, Maryland.

Cheng-Tao Chu, Sang Kyun Kim, Yi-An Lin, YuanYuan Yu, Gary Bradski, Andrew Ng, and Kunle Olukotun. 2006. Map-Reduce for machine learning on multicore. In *Advances in Neural Information Processing Systems 19 (NIPS 2006)*, pages 281–288, Vancouver, British Columbia, Canada.

Kenneth W. Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: Simplified data processing on large clusters. In *Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI 2004)*, pages 137–150, San Francisco, California.

Susan Dumais, Michele Banko, Eric Brill, Jimmy Lin, and Andrew Ng. 2002. Web question answering: Is more always better? In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002)*, pages 291–298, Tampere, Finland.

Chris Dyer, Aaron Cordova, Alex Mont, and Jimmy Lin. 2008. Fast, easy, and cheap: Construction of statistical machine translation models with MapReduce. In *Proceedings of the Third Workshop on Statistical Machine Translation at ACL 2008*, pages 199–207, Columbus, Ohio.

Tamer Elsayed, Jimmy Lin, and Douglas Oard. 2008. Pairwise document similarity in large collections with MapReduce. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL 2008), Companion Volume*, pages 265–268, Columbus, Ohio.

John R. Firth. 1957. A synopsis of linguistic theory 1930–55. In *Studies in Linguistic Analysis, Special Volume of the Philological Society*, pages 1–32. Blackwell, Oxford.

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. 2003. The Google File System. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP-03)*, pages 29–43, Bolton Landing, New York.

Zelig S. Harris. 1968. *Mathematical Structures of Language*. Wiley, New York.

Saif Mohammad and Graeme Hirst. 2006. Distributional measures of concept-distance: A task-oriented evaluation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, pages 35–43, Sydney, Australia.

Michael A. Rappa. 2004. The utility business model and the future of computing services. *IBM Systems Journal*, 34(1):32–42.

Hinrich Schütze and Jan O. Pedersen. 1998. A cooccurrence-based thesaurus and two applications to information retrieval. *Information Processing and Management*, 33(3):307–318.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123.

Jinxi Xu and W. Bruce Croft. 1998. Corpus-based stemming using cooccurrence of word variants. *ACM Transactions on Information Systems*, 16(1):61–81.

Online Acquisition of Japanese Unknown Morphemes using Morphological Constraints

Yugo Murawaki **Sadao Kurohashi**
Graduate School of Informatics, Kyoto University
Yoshida-honmachi, Sakyo-ku, Kyoto, 606-8501, Japan
murawaki@nlp.kuee.kyoto-u.ac.jp kuro@i.kyoto-u.ac.jp

Abstract

We propose a novel lexicon acquirer that works in concert with the morphological analyzer and has the ability to run in online mode. Every time a sentence is analyzed, it detects unknown morphemes, enumerates candidates and selects the best candidates by comparing multiple examples kept in the storage. When a morpheme is unambiguously selected, the lexicon acquirer updates the dictionary of the analyzer, and it will be used in subsequent analysis. We use the constraints of Japanese morphology and effectively reduce the number of examples required to acquire a morpheme. Experiments show that unknown morphemes were acquired with high accuracy and improved the quality of morphological analysis.

1 Introduction

Morphological analysis is the first step for most natural language processing applications. In Japanese morphological analysis, segmentation is processed simultaneously with the assignment of a part of speech (POS) tag to each morpheme. Segmentation is a nontrivial task in Japanese because it does not delimit words by white-space.

Japanese morphological analysis has successfully adopted dictionary-based approaches (Kurohashi et al., 1994; Asahara and Matsumoto, 2000; Kudo et al., 2004). In these approaches, a sentence is transformed into a lattice of morphemes by searching a pre-defined dictionary, and an optimal path in the lattice is selected.

This area of research may be considered almost completed, as previous studies reported the F-score of nearly 99% (Kudo et al., 2004). When applied to web texts, however, more errors are made due to unknown morphemes. In previous studies, experiments were performed on newspaper articles, but web texts include slang words, informal spelling alternates (Nishimura, 2003) and technical terms. For example, the verb "ググる" (*gugu-ru*, to google) is erroneously segmented into "ググ" (*gugu*) and "る" (*ru*).

One solution to this problem is to augment the lexicon of the morphological analyzer by extracting unknown morphemes from texts (Mori and Nagao, 1996). In the previous method, a morpheme extraction module worked independently of the morphological analyzer and ran in off-line (batch) mode. It is inefficient because almost all high-frequency morphemes have already been registered to the predefined dictionary. Moreover, it is inconvenient when applied to web texts because the web corpus is huge and diverse compared to newspaper corpora. It is not necessarily easy to build subcorpora before lexicon acquisition. Suppose that we want to analyze whaling-related documents. It is unnecessary and probably harmful to acquire morphemes that are irrelevant to the topic. A whaling-related subcorpus should be extracted from the whole corpus but it is not clear how large it must be.

We propose a novel lexicon acquirer that works in concert with the morphological analyzer and has the ability to run in online mode. As shown in Figure 1, every time a sentence is analyzed, the lexicon acquirer detects unknown morphemes, enumerates

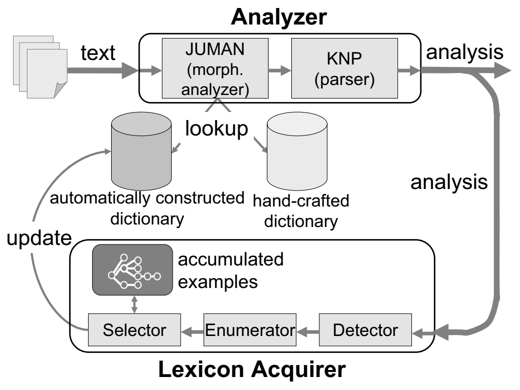


Figure 1: System architecture

candidates and selects the best candidates by comparing multiple examples kept in the storage. When a morpheme is unambiguously selected, the lexicon acquirer updates the automatically constructed dictionary, and it will be used in subsequent analysis. The proposed method is flexible and gives the system more control over the process. We do not have to limit the target corpus beforehand and the system can stop whenever appropriate.

We use the constraints of Japanese morphology that have already been coded in the morphological analyzer. These constraints effectively reduce the number of examples required to acquire an unknown morpheme. Experiments show that unknown morphemes were acquired with high accuracy and improved the quality of morphological analysis.

2 Japanese Morphology

In order to understand the task of lexicon acquisition, we briefly describe the Japanese morphological analyzer JUMAN.[1] We explain Japanese morphemes in Section 2.1, morphological constraints in Section 2.2, and unknown morpheme processing in Section 2.3.

2.1 Morpheme

In JUMAN, the POS tagset consists of four elements: class, subclass, conjugation type and conjugation form. The classes are noun, verb, adjective and others. Noun has subclasses such as common noun, *sa*-group noun, proper noun, organization, place, personal name. Verb and adjective have no subclasses.

Verbs and adjectives among others change their form according to the morphemes that occur after them, which is called conjugation. Conjugable morphemes are grouped by conjugation types such as vowel verb, *ra*-row verb, *i*-type adjective and *na*-type adjective. Each conjugable morpheme takes one of conjugation forms in texts. It has an invariant stem and an ending which changes according to conjugation type and conjugation form.

In this paper, the tuple of class, subclass and conjugation type is referred to as a POS tag. For simplicity, POS tags for nouns are called by their subclasses and those for verbs and adjectives by their conjugation types.

There are two types of morphemes: abstract dictionary entries, and examples or actual occurrences in texts. An entry consists of a stem and a POS tag while an example consists of a stem, a POS tag and a conjugation form. For example, the entry of the *ra*-row verb “走る” (*hashi-ru*, to run) can be represented as

(“走” (*hashi*), *ra*-row verb),

and their examples “走ら” (*hashi-ra*) and “走り” (*hashi-ri*) as

(“走” (*hashi*), *ra*-row verb, imperfective),

and

(“走” (*hashi*), *ra*-row verb, plain continuative)

respectively. As nouns do not conjugate, the entry of the *sa*-group noun “希望” (*kibou*, hope) can be represented as

(“希望” (*kibou*), *sa*-group noun)

and its sole example form is

(“希望” (*kibou*), *sa*-group noun, NIL).

2.2 Morphological Constraints

Japanese is an agglutinative language. Depending on its grammatical roles, a morpheme is followed by a sequence of grammatical suffixes, auxiliary verbs and particles, and the connectivity of these elements is bound by morphological constraints. For example, the particle “を” (*wo*, accusative case) can follow a verb with the conjugation form of plain continuative, as in “走りを” (*hashi-ri-wo*, running-ACC),

[1]`http://nlp.kuee.kyoto-u.ac.jp/nl-resource/juman.html`

but it cannot follow an imperfective verb ("*走らを" (*hashi-ra-wo*)).

These constraints are used by JUMAN to reduce the ambiguity. They can be also used in lexicon acquisition.

2.3 Unknown Morpheme Processing

Given a sentence, JUMAN builds a lattice of morphemes by searching a pre-defined dictionary, and then selects an optimal path in the lattice. To handle morphemes that cannot be found in the dictionary, JUMAN enumerates unknown morpheme candidates using character type-based heuristics, and adds them to the morpheme lattice. Unknown morphemes are given the special POS tag "undefined," which is treated as noun.

Character type-based heuristics are based on the fact that Japanese is written with several different character types such as kanji, hiragana and katakana, and that the choice of character types gives some clues on morpheme boundaries. For example, a sequence of katakana characters are considered as an unknown morpheme candidate, as in "グーグル" (*gûguru*, Google) out of "グーグルが" (*gûguru-ga*, Google-NOM). Kanji characters are segmented per character, which is sometimes wrong but prevents error propagation.

These heuristics are simple and effective, but far from perfect. They cannot identify mixed-character morphemes, verbs and adjectives correctly. For example, the verb "ググる" (*gugu-ru*, to google) is wrongly divided into the katakana unknown morpheme " ググ" (*gugu*) and the hiragana suffix "る" (*ru*).

3 Lexicon Acquisition

3.1 Task

The task of lexicon acquisition is to generate dictionary entries inductively from their examples in texts. Since the morphological analyzer provides a basic lexicon, the morphemes to be acquired are limited to those unknown to the analyzer.

In order to generate an entry, its stem and POS tag need to be identified. Determining the stem of an example is to draw the front and rear boundaries in a character sequence in texts which corresponds to the stem. The POS tag is selected from the tagset given by the morphological analyzer.

3.2 System Architecture

Figure 1 shows the system architecture. Each sentence in texts is processed by the morphological analyzer JUMAN and the dependency parser KNP.[2] JUMAN consults a hand-crafted dictionary and an automatically constructed dictionary. KNP is used to form a phrasal unit called *bunsetsu* by chunking morphemes.

Every time a sentence is analyzed, the lexicon acquirer receives the analysis. It detects examples of unknown morphemes and keeps them in storage. When an entry is unambiguously selected, the lexicon acquirer updates the automatically constructed dictionary, and it will be used in subsequent analysis.

3.3 Algorithm Overview

The process of lexicon acquisition has four phases: detection, candidate enumeration, aggregation and selection. First the analysis is scanned to detect examples of unknown morphemes. For each example, one or more candidates for dictionary entries are enumerated. It is added to the storage, and multiple examples in the storage that share the candidates are aggregated. They are compared and the best candidate is selected from it.

Take the *ra*-row verb "ググる" (*gugu-ru*) for example. Its example "ググってみた。" (*gugu-tte-mi-ta*, to have tried to google) can be interpreted in many ways as shown in Figure 2. Similarly, multiple candidates are enumerated for another example "ググるのは" (*gugu-ru-no-ha*, to google-TOPIC). If these examples are compared, we can see that the *ra*-row verb "ググる" (*gugu-ru*) can explain them.

3.4 Suffixes

Morphological constraints are used for candidate enumeration. Since they are coded in JUMAN, we first transform them into a set of strings called suffixes. A suffix is created by concatenating the ending of a morpheme (if any) and subsequent ancillary morphemes. Each POS tag is associated with a set of suffixes, as shown in Table 1. This means that a stem can be followed by one of the suffixes specified

[2]http://nlp.kuee.kyoto-u.ac.jp/nl-resource/knp.html

Table 1: Examples of suffixes

POS tag	base form	stem	ending	conjugation form[1]	suffixes
ra-row verb	*hashi-ru*	*hashi*	*ra*	imperfective	*razu*, *ranaide*
			ri	plain continuative	*riwo*, *riwomo*
			ru	plain	*ru*, *rukawo*
vowel verb	*akogare-ru*	*akogare*	ϕ	imperfective	*zu*, *naide*
			ϕ	plain continuative	*wo*, *womo*
			ru	plain	*ru*, *rukawo*
sa-group noun	*kibou*	*kibou*	NIL	*wo*	*wo*, *womo*
			NIL	*suru*	*suru*, *shitara*

[1] The conjugation form of a noun is substituted with the base form of its immediate ancillary morpheme because nouns do not conjugate.

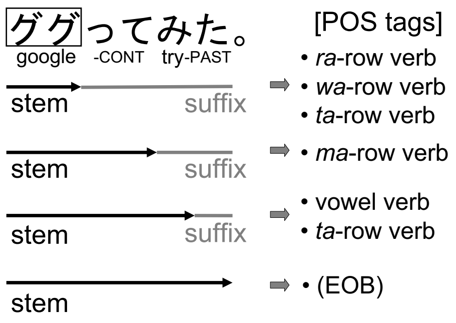


Figure 2: Candidate enumeration

by its POS tag and cannot be followed by any other suffix.

In preparation for lexicon acquisition, suffixes are acquired from a corpus. We used a web corpus that was compiled through the procedures proposed by Kawahara and Kurohashi (2006). Suffixes were extracted from examples of registered morphemes and were aggregated per POS tag.

We found that the number of suffixes did not converge even in this large-scale corpus. It was because ancillary morphemes included the wide variety of auxiliary verbs and formal nouns. Alternatively, we used the first five characters as a suffix. In the experiments, we obtained 500 thousand unique suffixes from 100 million pages. The number of POS tags that corresponded to a suffix was 1.33 on average.

3.5 Unknown Morpheme Detection

The first step of lexicon acquisition is unknown morpheme detection. Every time the analysis of a sentence was given, the sequence of morphemes are scanned, and suspicious points that probably represent unknown morphemes are detected.

Currently, we use the POS tag "undefined" to detect unknown morphemes. For example, the example "ググってみた。" is detected because "ググ" is given "undefined." This simple method cannot detect unknown morphemes if they are falsely segmented into combinations of registered morphemes. We leave the comprehensive detection of unknown morphemes to future work.

3.6 Candidate Enumeration

For each example, one or more candidates for the dictionary entry are enumerated. Each candidate is represented by a combination of a front boundary and the pair of a rear boundary and a POS tag.

The search range for enumeration is based on *bunsetsu* phrases, which is created by chunking morphemes. The range is at most the corresponding *bunsetsu* and the two immediately preceding and succeeding *bunsetsu*, which we found wide enough to contain correct candidates.

The candidates for the rear boundary and the POS tag are enumerated by string matching of suffixes as shown in Figure 2. If a suffix matches, the starting position of the suffix becomes a candidate for the rear boundary and the suffix is mapped to one or more corresponding POS tags.

In addition, the candidates for the front and rear boundaries are enumerated by scanning the sequence of morphemes. The boundary markers we use are

- punctuations,
- grammatical prefixes such as "御" (*go*-, honorific prefix), for front boundaries,

- grammatical suffixes such as “様” (*-sama*, honorific title), for rear boundaries, and
- *bunsetsu* boundaries given by KNP.

Each rear boundary candidate whose corresponding POS tag is not decided is given the special tag “EOB” (*end-of-bunsetsu*). This means that no suffix is attached to the candidate. Since nouns, vowel verbs and *na*-type adjectives can appear in isolation, it will be expanded to these POS tags when selecting the best POS tag.

3.7 Aggregation of Examples

Selection of the best candidate is done by comparing multiple examples. Each example is added to the storage, and then examples that possibly represent the same entry with it are extracted from the storage. Examples aggregated at this phase share the front boundary but may be unrelated to the example in question. They are pruned in the next phase.

In order to manage examples efficiently, we implement a trie. The example is added to the trie for each front boundary candidate. The key is the character sequence determined by the front boundary and the leftmost rear boundary. To retrieve examples that share the front boundary with it, we check every node in the path from the root to the node where it is stored, and collect examples stored in each node.

3.8 Selection

The best candidate is selected by identifying the front boundary, the rear boundary and the POS tag in this order. Starting from the rightmost front boundary candidate, multiple rear boundary candidates that share the front boundary are compared and some are dropped. Then starting from the leftmost surviving rear boundary candidate, the best POS tag is selected from the examples that share the stem. If the selected candidate satisfies simple termination conditions, it is added to the dictionary and the examples are removed from the storage.

For each front boundary candidate, some inappropriate rear boundary candidates are dropped by examining the inclusion relation between the examples of a pair of candidates. The assumption behind this is that an appropriate candidate can interpret more examples than incorrect ones. Let p and q be a pair of the candidates for the rear boundary, and R_p and R_q be the sets of examples for which p and q are enumerated. If p is a prefix of q and p is the correct stem, then R_q must be contained in R_p. In practice we loosen this condition, considering possible errors in candidate enumeration

For each stem candidate, the appropriate POS tag is identified. Similarly to rear boundary identification, POS identification is done by checking inclusion relation.

If the POS tag is successfully disambiguated, simple termination conditions is checked to prevent the accidental acquisition of erroneous candidates. The first condition is that the number of unique conjugation forms that appear in the examples should be 3 or more. If the candidate is a noun, it is substituted with the number of the unique base forms of their immediate ancillary morphemes. The second condition is that the front boundaries of some examples are decided by clear boundary markers such as punctuations and the beginning of sentence. This prevents oversegmentation. For example, the stem candidate “*撰組” (**sengumi*) is always enumerated for examples of “新撰組” (*Shingengumi*, a historical organization) since “新” (*shin-*, new) is a prefix. This candidate is not acquired because “*撰組” (**sengumi*) does not occur alone and is always accompanied by “新” (*shin-*). Thresholds are chosen empirically.

3.9 Decompositionality

Since a morpheme is extracted from a small number of examples, it is inherently possible that the acquired morpheme actually consists of two or more morphemes. For example, the noun phrase “顆粒タイプ” (*karyuu-taipu*, granular type) may be acquired as a morpheme before “顆粒” (*karyuu*, granule) is extracted. To handle this phenomenon, it is checked at the time of acquisition whether the new morpheme (*kairyuu*) can decompose registered morphemes (*kairyuu-taipu*). If found, a composite “morpheme” is removed from the dictionary.

Currently we leave the decompositionality check to the morphological analyzer. Possible compounds are enumerated by string matching and temporarily removed from the dictionary. Each candidate is analyzed by the morphological analyzer and it is checked whether the candidate is divided into a combination of registered morphemes. If not, the candidate is restored to the dictionary.

Table 2: Statistical information per query

query	number of sentences	number of affected sentences (ratio)	number of acquired morphs	number of correct morphs (precision)	number of examples[1]
捕鯨問題 (whaling issue)	135,379	2,444 (1.81%)	293	290 (99.0%)	4
赤ちゃんポスト (baby hatch)	74,572	775 (1.04%)	107	105 (98.1%)	4
ジャスラック (JASRAC)	195,928	6,259 (3.19%)	913	907 (99.3%)	4
ツンデレ (tsundere)	77,962	12,012 (15.4%)	243	238 (97.4%)	5
アガリクス (agaricus)	78,922	3,037 (3.85%)	114	107 (93.9%)	9

[1] The median number of examples used for acquisition.

4 Experiments

4.1 Experimental Design

We used the default dictionary of the morphological analyzer JUMAN as the initial lexicon. It contained 30 thousand basic morphemes. If spelling variants were expanded and proper nouns were counted, the total number of morphemes was 120 thousands.

We used domain-specific corpora as target texts because efficient acquisition was expected. If target texts shared a topic, relevant unknown morphemes were used frequently. In the experiments, we used search engine TSUBAKI (Shinzato et al., 2008) and casted the search results as domain-specific corpora. For each query, our system sequentially read pages from the top of the result and acquired morphemes. We terminated the acquisition at the 1000th page and analyzed the same 1000 pages with the augmented lexicon. The queries used were "捕鯨問題" (whaling issue), "赤ちゃんポスト" (baby hatch), "ジャスラック" (JASRAC, a copyright collective), "ツンデレ" (tsundere, a slang word) and "アガリクス" (agaricus).

4.2 Evaluation Measures

The proposed method is evaluated by measuring the accuracy of acquired morphemes and their contribution to the improvement of morphological analysis. A morpheme is considered accurate if both segmentation and the POS tag are correct. Note that segmentation is a nontrivial problem for evaluation. In fact, the disagreement over segmentation criteria was considered one of the main reasons for reported errors by Nagata (1999) and Uchimoto et al. (2001). It is difficult to judge whether a compound term should be divided because there is no definite standard for morpheme boundaries in Japanese. For example, "ミンク鯨" (*minku-kujira*, minke whale) can be extracted as a single morpheme or decomposed into "ミンク" and "鯨." While segmentation is an open question in Japanese morphological analysis, "correct" segmentation is not necessarily important for applications using morphological analysis. Even if a noun is split into two or more morphemes in morphological analysis, they are chunked to form a phrasal unit called *bunsetsu* in dependency parsing, and to extract a keyword (Nakagawa and Mori, 2002).

To avoid the decompositionality problem, we adopted manual evaluation. We analyzed the target texts with both the initial lexicon and the augmented lexicon. Then we checked differences between the two analyses and extracted sentences that were affected by the augmentation. Among these sentences, we evaluated randomly selected 50 sentences per query. We checked the accuracy of segmentation and POS tagging of each "diff" block, which is illustrated in Figure 3. The segmentation of a block was judged correct unless morpheme boundaries were clearly wrong.

In the evaluation of POS tagging, we did not distinguish subclasses of noun[3] such as common noun

[3]In the experiments, we regarded demonstrative pronouns as

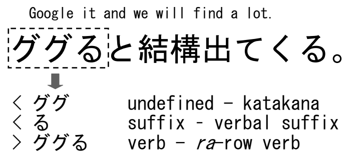
Table 3: Examples of acquired morphemes

query	examples
whaling issue	モラトリアム (moratorium), ツチクジラ (giant beaked whale), 混獲 (bycatch)
baby hatch	ダンナ (husband), 助産師 (midwife), 棄てる (to abandon), 訊く (to inquire)
JASRAC	ソフ倫 (an organization), シャ乱 Q (a pop-rock band), ヲタ (geek)
tsundere	アキバ (abbr. of Akihabara), 腐女子 (*fujoshi*, a slang word), モテる (to be popular)
agaricus	サプリ (abbr. of suppliment), アロマ (aroma), 食効 (enhanced nutritional function)

Table 4: Evaluation of "diff" blocks

	segmentation				POS tagging				
query	E → C	C → C	E → E	C → E	E → C	C → C	E → E	C → E	total
whaling issue	11	**45**	0	2	11	**45**	0	2	58
baby hatch	**37**	12	0	3	**37**	12	0	3	52
JASRAC	16	**23**	1	12	16	**23**	1	12	52
tsundere	17	**39**	0	1	17	**39**	0	1	57
agaricus	22	**31**	0	0	22	**31**	0	0	53

(Legend – C: correct; E: erroneous)



```
< ググ      undefined - katakana
< る        suffix - verbal suffix
> ググる    verb - ra-row verb
```

Figure 3: A "diff" block in a sentence

and proper noun. The special POS tag "undefined" given by JUMAN was treated as noun.

4.3 Results

Table 2 summarizes statistical information per query. The number of sentences affected by the augmentation varied considerably (1.04%–15.4%). The initial lexicon of the morphological analyzer lacked morphemes that appeared frequently in some corpora because morphological analysis had been tested mainly with newspaper articles.

The precision of acquired morphemes was high (97.4%–99.3%), and the number of examples used for acquisition was as little as 4–9. These results are astonishing considering that Mori and Nagao (1996) ignored candidates that appeared less than 10 times (because they were unreliable).

nouns because their morphological behaviors were the same as those of nouns. Although demonstrative nouns are closed class morphemes, their katakana forms such as "コレ" (this) were acquired as nouns. The morphological analyzer assumed that demonstrative pronouns were written in hiragana, e.g., "これ," as they always are in a newspaper.

Table 3 shows some acquired morphemes. As expected, the overwhelming majority were nouns (93.0%–100%) and katakana morphemes (80.7%–91.6%). Some were mixed-character morphemes ("ソフ倫" and "シャ乱 Q"), which cannot be recognized by character-type based heuristics, and slang words ("腐女子," "ヲタ," etc.) which did not appear in newspaper articles. Some morphemes were spelling variants of those in the pre-defined dictionary. Uncommon kanji characters were used in basic words ("棄てる" for "捨てる" and "訊く" for "聞く") and katakana was used to change nuances ("モテる" for "もてる" and "ダンナ" for "旦那").

Table 4 shows the results of manual evaluation of "diff" blocks. The overwhelming majority of blocks were correctly analyzed with the augmented lexicon (E → C and C → C). On the other hand, adverse effects were observed only in a few blocks (C → E). In conclusion, acquired morphemes improve the quality of morphological analysis.

4.4 Error Analysis

Some short katakana morphemes oversegmented other katakana nouns. For example, "サーバー" (*sâbâ*, server) was wrongly segmented by newly-acquired "サー" (sâ, sir) and preregistered "バー" (*bâ*, bar). Neither the morphological analyzer and the lexicon acquirer could detect this semantic mismatch. Curiously, one example of "サー" (sâ) was actuallly part of "サーバー" (*sâbâ*), which was erro-

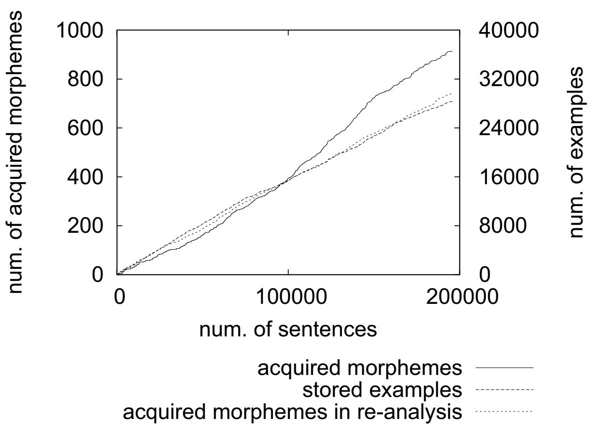


Figure 4: Process of online acquisition

neously segmented when extracting sentences from HTML.

The katakana adjective "イイ" (*i-i*, good), a spelling variant of the basic morpheme "いい," was falsely identified as a noun because its ending "イ" was written in katakana. The morphological analyzer, and hence the lexicon acquirer, assume that the ending of a verb or adjective is written in hiragana. This assumption is reasonable for standard Japanese, but does not always hold when we analyze web texts. In order to recognize unconventional spellings that are widely used in web texts (Nishimura, 2003), more flexible analysis is needed.

4.5 Discussion

It is too costly or impractical to calculate the recall of acquisition, or the ratio of the number of acquired morphemes against the total number of unknown morphemes because it requires human judges to find *undetected* unknown morphemes from a large amount of raw texts.

Alternatively, we examined the ratio against the number of *detected* unknown morphemes. Figure 4 shows the process of online acquisition for the query "JASRAC." The monotonic increase of the numbers of acquired morphemes and stored examples suggests that the vocabulary size did not converge. The number of occurrences of acquired morphemes in re-analysis was approximately the same with the number of examples kept in the storage during acquisition. This means that, in terms of frequency of occurrence, about half of unknown morphemes were acquired. Most unknown morphemes belong to the "long tail" and the proposed method seems to have seized a "head" of the long tail.

Although some previous studies emphasized correct identification of low frequency terms (Nagata, 1999; Asahara and Matsumoto, 2004), it is no longer necessary because very large scale web texts are available today. If a small set of texts needs to be analyzed with high accuracy, we can incorporate similar texts retrieved from the web, to increase the number of examples of unknown morphemes. The proposed method can be modified to check if unknown morphemes detected in the initial set are acquired and to terminate whenever sufficient acquisition coverage is achieved.

5 Related Work

Since most languages delimit words by white-space, morphological analysis in these languages is to segment words into morphemes. For example, Morpho Challenge 2007 (Kurimo et al., 2007) was evaluations of unsupervised segmentation for English, Finnish, German and Turkish.

While Japanese is an agglutinative language, other non-segmented languages such as Chinese and Thai are analytic languages. Among them, Chinese has been a subject of intensive research. Peng et al. (2004) integrated new word detection into word segmentation. They detected new words by computing segment confidence and re-analyzed the inputs with detected words as features.

The Japanese language is unique in that it is written with several different character types. Heuristics widely used in unknown morpheme processing are based on character types. They were also used as important clues in statistical methods. Nagata (1999) integrated a probabilistic unknown word models into the word segmentation model. Uchimoto et al. (2001) incorporated them as feature functions of a Maximum Entropy-based morphological analyzer. Asahara and Matsumoto (2004) used them as a feature of character-based chunking of unknown words using Support Vector Machines.

Mori (1996) extracted words from texts and estimated their POSs using distributional analysis. The appropriateness of a word candidate was measured by the distance between probability distributions of the candidate and a model. In this method, morphological constraints were indirectly represented by distributions.

Nakagawa and Matsumoto (2006) presented a method for guessing POS tags of pre-segmented unknown words that took into consideration all the occurrences of each unknown word in a document. This setting is impractical in Japanese because POS tagging is inseparable from segmentation.

6 Conclusion

We propose a novel method that augments the lexicon of a Japanese morphological analyzer by acquiring unknown morphemes from texts in online mode. Unknown morphemes are acquired with high accuracy and improve the quality of morphological analysis.

Unknown morphemes are one of the main sources of error in morphological analysis when we analyze web texts. The proposed method has the potential to overcome the unknown morpheme problem, but it cannot be achieved without recognizing or being robust over various phenomena such as unconventional spellings and typos. These phenomena are not observed in newspaper articles but cannot be ignored in web texts. In the future, we will work on these phenomena.

Morphological analysis is now very mature. It is widely applied as preprocessing for NLP applications such as parsing and information retrieval. Hence in the future, we aim to use the proposed method to improve the quality of these applications.

References

Masayuki Asahara and Yuji Matsumoto. 2000. Extended models and tools for high-performance part-of-speech tagger. In *Procs. of COLING 2000*, pages 21–27.

Masayuki Asahara and Yuji Matsumoto. 2004. Japanese unknown word identification by character-based chunking. In *Procs. of COLING 2004*, pages 459–465.

Daisuke Kawahara and Sadao Kurohashi. 2006. Case frame compilation from the web using high-performance computing. In *Procs. of LREC-06*, pages 1344–1347.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Procs. of EMNLP 2004*, pages 230–237.

Mikko Kurimo, Mathias Creutz, and Ville Turunen. 2007. Overview of Morpho Challenge in CLEF 2007. In *Working Notes of the CLEF 2007 Workshop*, pages 19–21.

Sadao Kurohashi, Toshihisa Nakamura, Yuji Matsumoto, and Makoto Nagao. 1994. Improvements of Japanese morphological analyzer JUMAN. In *Procs. of The International Workshop on Sharable Natural Language Resources*, pages 22–38.

Shinsuke Mori and Makoto Nagao. 1996. Word extraction from corpora and its part-of-speech estimation using distributional analysis. In *Procs. of COLING 1996*, pages 1119–1122.

Masaaki Nagata. 1999. A part of speech estimation method for Japanese unknown words using a statistical model of morphology and context. In *Procs. of ACL 1999*, pages 277–284.

Tetsuji Nakagawa and Yuji Matsumoto. 2006. Guessing parts-of-speech of unknown words using global information. In *Procs. of COLING-ACL 2006*, pages 705–712.

Hiroshi Nakagawa and Tatsunori Mori. 2002. A simple but powerful automatic term extraction method. In *COLING-02 on COMPUTERM 2002*, pages 29–35.

Yukiko Nishimura. 2003. Linguistic innovations and interactional features of casual online communication in Japanese. *Journal of Computer-Mediated Communication*, 9(1).

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *Procs. of COLING '04*, pages 562–568.

Keiji Shinzato, Tomohide Shibata, Daisuke Kawahara, Chikara Hashimoto, and Sadao Kurohashi. 2008. TSUBAKI: An open search engine infrastructure for developing new information access methodology. In *Procs. of IJCNLP-08*, pages 189–196.

Kiyotaka Uchimoto, Satoshi Sekine, and Hitoshi Isahara. 2001. The unknown word problem: a morphological analysis of Japanese using maximum entropy aided by a dictionary. In *Procs. of EMNLP 2001*, pages 91–99.

Legal Docket-Entry Classification: Where Machine Learning stumbles

Ramesh Nallapati and Christopher D. Manning
Natural Language Processing Group
Department of Computer Science
Stanford University
Stanford, CA 94305
{nmramesh,manning}@cs.stanford.edu

Abstract

We investigate the problem of binary text classification in the domain of legal docket entries. This work presents an illustrative instance of a domain-specific problem where the state-of-the-art Machine Learning (ML) classifiers such as SVMs are inadequate. Our investigation into the reasons for the failure of these classifiers revealed two types of prominent errors which we call conjunctive and disjunctive errors. We developed simple heuristics to address one of these error types and improve the performance of the SVMs. Based on the intuition gained from our experiments, we also developed a simple propositional logic based classifier using hand-labeled features, that addresses both types of errors simultaneously. We show that this new, but simple, approach outperforms all existing state-of-the-art ML models, with statistically significant gains. We hope this work serves as a motivating example of the need to build more expressive classifiers beyond the standard model classes, and to address text classification problems in such non-traditional domains.

1 Introduction

Text Classification is a widely researched area, with publications spanning more than a decade (Yang and Liu, 1999). Although earlier models used logic based rules (Apté et al., 1994) and decision trees (Lewis and Ringuette, 1994), recently the emphasis has been on statistical classifiers such as the naive Bayes model (McCallum and Nigam, 1998), logistic regression (Zhang and Oles, 2001) and support vector machines (Joachims, 1998). Although several complex features were considered for classification, eventually researchers have settled down to simple bag-of-words features such as unigrams and some times bigrams (Dumais et al., 1998), thereby completely ignoring the grammar and other semantic information in the text. Despite this fact, the state-of-the-art performance is close to or above 90% on F1 scores on most standard test collections such as Reuters, 20 newsgroups, *etc.* (Bekkerman et al., 2003). As such, most researchers and practitioners believe text classification technology has reached a mature state, where it is suitable for deployment in real life applications.

In this work, we present a text classification problem from the legal domain which challenges some of our understanding of text classification problems. In the new domain, we found that the standard ML approaches using bag-of-words features perform relatively poorly. Not only that, we noticed that the linear form (or even polynomial form) used by these classifiers is inadequate to capture the semantics of the text. Our investigation into the shortcomings of the traditional models such as SVMs, lead us to build a simple propositional logic based classifier using hand-labeled features that outperforms these strong baselines.

Although the new model by itself is interesting, the main objective of our work is to present the text classification community with an interesting problem where the current models are found inadequate. Our hope is that the new problem will encourage researchers to continue to build more sophisticated models to solve classification problems in diverse, non-traditional domains.

The rest of the paper is organized as follows. In section 2, we introduce the problem of legal docket entry classification and describe the data with some representative examples. In section 3, we describe the experiments performed with SVMs and several of its variants. We also identify the shortcomings of the current classifiers in this section. In section 3.2, we present results from using human selected features for the classification problem and motivate their application for the docket entry classification using propositional logic in subsection 3.3. We also show that simple propositional logic using human selected features and their labels outperforms the state-of-the-art classifiers. We conclude the discussion in section 4, where we argue the case for more sophisticated classifiers for specialized domains.

2 Docket Entry Classification

In this section, we introduce the problem of legal docket entry classification.

In any US district court of law, information on the chronological events in a case is usually entered in a document called the *case docket*. Each entry in a docket lists an event that occured on a specific date such as pleading, appeal, order, jury trial, judgment, etc. The entries are brief descriptions of the events in natural language. Sometimes, a single docket entry can list multiple events that take place on the same day. Table 1 displays a sample docket for a case.

Identifying various events in a court case is a crucial first step to automatically understanding the progression of a case and also in gathering aggregate statistics of court cases for further analysis. While some events such as "Complaint" may be easy to identify using regular expressions, others are much more complex and may require sophisticated modeling.

In this work, we are primarily interested in identifying one such complex event called "Order re: Summary Judgment". Summary Judgment is a legal term which means that a court has made a determination (a judgment) without a full trial.[1] Such a judgment may be issued as to the merits of an entire case, or of specific issues in that case. Typically, one of the parties (plaintiff or defendant) involved in the case moves a motion for summary judgment, (usually) in an attempt to eliminate the risk of losing a trial. In an "Order re: Summary Judgment" event, the court may grant or deny a motion for summary judgment upon inspecting all the evidence and facts in the case. The task then, is to identify all docket entries in a set of cases that list occurrences of "Order re: Summary Judgment" events. We will call them OSJ events in short.

A few typical positive and negative docket entries for the OSJ event from various cases are shown in table 2. The examples require some explanation. Firstly, all orders granting, denying or amending motions for full or partial summary judgment are considered OSJs. However, if the motion is denied as moot or denied without prejudice, it is not an OSJ event, as shown in the negative examples 1 and 2 in table 2. This is because in such cases, no decision was made on substantive issues of the case. Also, there are other kinds of orders that are issued with reference to a summary judgment motion that do not fall into the category of OSJ, such as negative examples 3 through 9. To elaborate further, negative example 3 is about amending the deadline for filing a summary judgment motion, but not a summary judgment motion itself. Likewise, in negative example 4, the judge denies a motion to shorten time on a motion to vacate the order on summary judgment, but not the motion on summary judgment itself. The other negative examples are very similar in spirit and we leave it as an exercise to the reader to interpret why they are negatively labeled.

On first glance, it appears that a standard classifier may do a good job on this data, since the classification seems to depend mostly on certain key words such as 'granting', 'denying', 'moot', etc. Also notice that some of the docket entries contain multiple events, but as long as it contains the 'order re: summary judgment' event, it falls into the positive class. This seems very similar to the standard case, where a document may belong to multiple topics, but it is still identified as on-topic by a binary classifier on the corresponding topic.

Hence, as a first step, we attempted using a standard SVM classifier.

[1]See *e.g.,* Wikipedia for more information: http://en.wikipedia.org/wiki/Summary_judgment

#	Date Filed	Text
1	10/21/2002	Original Complaint with JURY DEMAND filed. Cause: 35:271 Patent Infringement Modified on 10/24/2002 (Entered: 10/22/2002)
2	10/21/2002	Form mailed to Commissioner of Patents and Trademarks. (poa)
3	10/28/2002	Return of service executed as to Mathworks Inc 10/23/02 Answer due on 11/12/02 for Mathworks Inc (poa) (Entered: 10/28/2002)
4	11/4/2002	Unopposed Motion by Mathworks Inc The to extend time to answer or otherwise respond to pla's complaint (ktd) (Entered: 11/05/2002)
5	11/5/2002	ORDER granting [4-1] motion to extend time to answer or otherwise respond to pla's complaint, ans reset answer due on 11/27/02 for Mathworks Inc
...	...	......

Table 1: An example (incomplete) docket: each row in the table corresponds to a docket-entry

2.1 Data

We have collected 5,595 docket entries from several court cases on intellectual property litigation, that are related to orders pertaining to summary judgment, and hand labeled them into OSJ or not OSJ categories.[2] The hand-labeling was done by a single legal expert, who practised law for a number of years. In all, 1,848 of these docket entries fall into the OSJ category.

In all our experiments, we split the entire data randomly into 20 disjoint subsets, where each set has the same proportion of positive-to-negative examples as the original complete set. For all the classifiers we used in this work, we performed 20-fold cross validation. We compute F1 scores on the held-out data of each run and report overall F1 score as the single point performance measure. We also perform statistical significance tests using the results from the 20 cross-validation runs.

2.2 Preprocessing

Before we ran our classifiers, we removed all punctuation, did casefolding, removed stopwords and stemmed the words using the Porter stemmer. We used unigrams and bigrams as our basic features.[3] We considered all the words and bigrams as binary features and did not use any TF-IDF weighting. Our justification for this decision is as follows: the docket text is typically very short and it is usually rare to see the same feature occurring multiple times in a docket entry. In addition, unlike in standard text classification, some of the features that are highly frequent across docket entries such as 'denying','granting', etc., are also the ones that are highly discriminative. In such a case, down-weighting these features using IDF weights might actually hurt performance. Besides (Dumais et al., 1998) found that using binary features works as well as using TF-IDF weights.

In addition, we also built a domain specific sentence boundary detector using regular expressions.[4] For constructing the features of a docket entry, we only consider those sentences in the entry that contain the phrase "summary judgment" and its variants.[5] Our preliminary experiments found that this helps the classifier focus on the relevant features, helping it to improve precision while not altering its recall noticeably.

3 Experiments and results

3.1 Basic SVM

First we implemented the standard linear SVM[6] on this problem with only word-based features (unigrams and bigrams) as the input. Quite surprisingly, the model achieves an F1 score of only 79.44% as shown in entry 1 of table 5. On inspection, we no-

[2]The data can be made available free of cost upon request. Please email the first author for more information.

[3]In our preliminary experiments, we found that a combination of unigrams and bigrams works better than unigrams alone.

[4]It works well in most cases but is far from perfect, due to the noisy nature of the data.

[5]The variants include "sum jgm", "S/J", "summary adjudication", "summary jgm", etc.

[6]All our SVM experiments were performed using the libsvm implementation downloadable from http://www.csie.ntu.edu.tw/∼cjlin/libsvm/

REPRESENTATIVE POSITIVE EXAMPLES

1. ORDER denying [36-1] motion for summary judgment on dfts Ranbaxy invalidity defenses by pltfs. (signed by Judge Garrett E. Brown, Jr.)
2. ORDER GRANTING IN PART AND DENYING IN PART DEFENDANTS' MOTION FOR SUMMARY JUDGMENT
3. ORDER re 78 MOTION to Amend/Correct Motion for Summary Judgment and supporting documents, filed by Defendant Synergetics USA, Inc. ; ORDERED GRANTED.
4. MEMORANDUM AND ORDER re: 495 Third MOTION for Partial Summary Judgment Dismissing Monsanto's Defenses Related to Dr. Barnes filed by Bayer BioScience N.V., motion is GRANTED IN PART AND DENIED IN PART.
5. ORDER GRANTING IN PART PLTF S/J MOT; GRANTING IN PART PLTF MOT/CLARIFY; GRANTING DEFT MOT/CLARIFY; PRTL S/J STAYED.
6. ORDER by Chief Judge Joe B. McDade. Court is granting in part and denying in part Deere's motion for reconsideration and clarification [42-2]; granting Toro's motion for summary judgment of non-infringement [45-1]; denying Deere's motion for summary judgment [58-1];
7. ORDER GRANTING DEFT. MOTION FOR S/J AND DENYING PLTF. MOTIONS FOR S/J AND TO SUPPLEMENT.

REPRESENTATIVE NEGATIVE EXAMPLES

1. ORDER - denying w/out prejudice 17 Motion for Summary Judgment, denying w/out prejudice 49 Motion to Amend/Correct . Signed by Judge Kent A. Jordan on 1/23/06.
2. Order denying as moot motion for summary judgment.
3. Order granting 53 Motion to Amend/Correct the deadline for filing summary jgm motions will be moved 12/1/03 to 12/8/03
4. ORDER by Judge Claudia Wilken denying plaintiff's motion to shorten time on motion to vacate portions of Court's order on cross-motion for summary judgment on patent issues [695-1] [697-1]
5. MEMORANDUM AND ORDER: by Honorable E. Richard Webber, IT IS HEREBY ORDERED that Defendant Aventis shall have 10 days from the date of this order to demonstrate why the Court should not grant summary judgment to Monsanto of non-infringement of claims 1-8 and 12 of the '565 patent and claim 4 of the '372 patent.
6. ORDER by Judge Claudia Wilken DENYING motion for an order certifying for immediate appeal portions of the courts' 2/6/03 order granting in part plaintiff's motion for partial summary judgment [370-1]
7. ORDER by Judge William Alsup denying in part 12 Motion to Consolidate Cases except as to one issue, granting in part for collateral estoppel 20 Motion for Summary Judgment
8. ORDER (Chief Mag. Judge Jonathan G. Lebedoff / 9/11/02) that the court grants Andersen's motion and orders that Andersen be allowed to bring its motions for summary judgment
9. ORDER by Judge Susan J. Dlott denying motion to strike declaration of H Bradley Hammond attached to memorandum in opposition to motion for partial summary judgment as to liability on the patent infringement and validity claims [40-1] [47-1] [48-1]

Table 2: Order: re Summary Judgment: positive and negative docket entries. The entries are reproduced as they are.

ticed that the SVM assigns high weights to many spurious features owing to their strong correlation with the class.

As a natural solution to this problem, we selected the top 100 features[7] using the standard information gain metric (Yang and Pedersen, 1997) and ran the SVM on the pruned feature set. As one would expect, the performance of the SVM improved significantly to reach an F1 score of 83.08% as shown in entry 2 of the same table. However, it is still a far cry from the typical results on standard test beds where the performance is above 90% F1. We suspected that training data was probably insufficient, but a learning curve plotting performance of the SVM as a function of the amount of training data reached a plateau with the amount of training data we had, so this problem was ruled out.

To understand the reasons for its inferior performance, we studied the features that are assigned the highest weights by the classifier. Although the SVM is able to assign high weights to several discriminative features such as ‘denied’, and ‘granted’, it also assigns high weights to features such as ‘opinion’, ‘memorandum’, ‘order’, ‘judgment’, etc., which have high co-occurrence rates with the positive class, but are not very discriminative in terms of the actual classification.

This is indicative of the problems associated with standard feature selection algorithms such as information gain in these domains, where high correlation with the label does not necessarily imply high discriminative power of the feature. Traditional classification tasks usually fall into what we call the ‘topical classification’ domain, where the distribution of words in the documents is a highly discriminative feature. On such tasks, feature selection algorithms based on feature-class correlation have been very successful. In contrast, in the current problem, which we call ‘semantic classification’, there seem to be a fixed number of domain specific operative words such as ‘grant’, ‘deny’, ‘moot’, ‘strike’, etc., which, almost entirely decide the class of the docket entry, irrespective of the existence of other highly correlated features. The information gain metric as well as the SVM are not able to fully capture such features in this problem.

[7]We tried other numbers as well, but top 100 features achieves the best performance.

We leave the problem of accurate feature selection to future work, but in this work, we address the issue by asking for human intervention, as we describe in the next section. One reason for seeking human assistance is that it will give us an estimate of upperbound performance of an automatic feature selection system. In addition, it will also offer us a hint as to whether the poor performance of the SVM is because of poor feature selection. We will aim to answer this question in the next section.

3.2 Human feature selection

Using human assistance for feature selection is a relatively new idea in the text classification domain. (Raghavan et al., 2006) propose a framework in which the system asks the user to label documents and features alternatively. They report that this results in substantial improvement in performance especially when the amount of labeled data is meagre. (Druck et al., 2008) propose a new Generalized Expectation criterion that learns a classification function from labeled features alone (and no labeled documents). They showed that feature labeling can reduce annotation effort from humans compared to document labeling, while achieving almost the same performance.

Following this literature, we asked our annotators to identify a minimal but definitive list of discriminative features from labeled data. The annotators were specifically instructed to identify the features that are most critical in tagging a docket entry one way or the other. In addition, they were also asked to assign a polarity to each feature. In other words, the polarity tells us whether or not the features belong to the positive class. Table 3 lists the complete set of features identified by the annotators.

As an obvious next step, we trained the SVM in the standard way, but using only the features from table 3 as the pruned set of features. Remarkably, the performance improves to 86.77% in F1, as shown in entry 3 of table 5. Again, this illustrates the uniqueness of this dataset, where a small number of hand selected features (< 40) makes a huge difference in performance compared to a state-of-the-art SVM combined with automatic feature selection. We believe this calls for more future work in improving feature selection algorithms.

Label	Features
Positive	grant, deny, amend, reverse, adopt, correct, reconsider, dismiss
Negative	strike, proposed, defer, adjourn, moot, exclude, change, extend, leave, exceed, premature, unseal, hearing, extend, permission, oral argument, schedule, ex parte, protective order, oppose, without prejudice, withdraw, response, suspend, request, case management order, to file, enlarge, reset, supplement placing under seal, show cause reallocate, taken under submission

Table 3: Complete set of hand-selected features: morphological variants not listed

Notice that despite using human assistance, the performance of the SVM is still not at a desirable level. This clearly points to deficiencies in the model other than poor feature selection. To understand the problem, we examined the errors made by the SVM and found that there are essentially two types of errors: *conjunctive* and *disjunctive*. Representative examples for both kinds of errors are displayed in table 4. The first example in the table corresponds to a conjunctive error, where the SVM is unable to model the binary switch like behavior of features. In this example, although 'deny' is rightly assigned a positive weight and 'moot' is rightly assigned a negative weight, when both features co-occur in a docket entry (as in 'deny as moot'), it makes the label negative.[8] However, the combined weight of the linear SVM is positive since the absolute value of the weight assigned to 'deny' is higher than that of 'moot', resulting in a net positive score. The second example falls into the category of disjunctive errors, where the SVM fails to model disjunctive behavior of sentences. In this example, the first sentence contains an OSJ event, but the second and third sentences are negatives for OSJ. As we have discussed earlier, this docket entry belongs to the OSJ category since it contains at least one OSJ event. However, we see that the negative weights assigned by the SVM to the second and third sentences result in an overall negative classification.

As a first attempt, we tried to reduce the conjunctive errors in our system. Towards this objective, we built a decision tree[9] using the same features listed in table 3. Our intuition was that a decision tree makes a categorical decision at each node in the tree, hence it could capture the binary-switch like behavior of features. However, the performance of the decision tree is found to be statistically indistinguishable from the linear SVM as shown in entry 4 of table 5. As an alternative, we used an SVM with a quadratic kernel, since it can also capture such pairwise interactions of features. This resulted in a fractional improvement in performance, but is again statistically indistinguishable from the decision tree. We also tried higher order polynomial kernels and the RBF kernel, but the performance got no better.[10] It is not easy to analyze the behavior of non-linear kernels since they operate in a higher kernel space. Our hypothesis is that polynomial functions capture higher order interactions between features, but they do not capture conjunctive behavior precisely.

As an alternative, we considered the following heuristic: whenever two or more of the hand selected features occur in the same sentence, we merged them to form an n-gram. The intuition behind this heuristic is the following: using the same example as before, if words such as 'deny' and 'moot' occur in the same sentence, we form the bigram 'deny-moot', forcing the SVM to consider the bigram as a separate feature. We hope to capture the conjunctive behavior of some features using this heuristic. The result of this approach, as displayed in entry 6 of table 5, shows small but statistically significant improvement over the quadratic SVM, confirming our theory. We also attempted a quadratic kernel using sentence level n-grams, but it did not show any improvement.

Note that all the models and heuristics we used above only address conjunctive errors, but not disjunctive errors. From the discussion above, we suspect the reader already has a good picture of what

[8]This is very similar to the conjunction of two logical variables where the conjunction of the variables is negative when at least one of them is negative. Hence the name conjunctive error.

[9]We used the publicly available implementation from www.run.montefiore.ulg.ac.be/∼francois/software/jaDTi/

[10]We also tried various parameter settings for these kernels with no success.

1. DOCKET ENTRY: order denying as moot [22-1] motion for summary judgment (signed by judge federico a. moreno on 02/28/06).
   FEATURES (WEIGHTS): denying (1.907), moot (-1.475)
   SCORE: 0.432; TRUE LABEL: Not OSJ; SVM LABEL: OSJ
2. DOCKET ENTRY: order granting dfts' 37 motion for summary judgment. further ordered denying as moot pla's cross-motion 42 for summary judgment. denying as moot dfts' motion to strike pla's cross-motion for summary judgment 55 . directing the clerk to enter judgment accordingly. signed by judge mary h murguia on 9/18/07
   FEATURES (WEIGHTS): granting (1.64), denying (3.57), strike(-2.05) moot(-4.22)
   SCORE: -1.06; TRUE LABEL: OSJ; SVM LABEL: Not OSJ

Table 4: Representative examples for conjunctive and disjunctive errors of the linear SVM using hand selected features

an appropriate model for this data might look like. The next section introduces this new model developed using the intuition gained above.

3.3 Propositional Logic using Human Features and Labels

So far, the classifiers we considered received a performance boost by piggybacking on the human selected features. However, they did not take into account the polarity of these features. A logical next step would be to exploit this information as well. An appropriate model would be the generalized expectation criterion model by (Druck et al., 2008) which learns by matching model specific label expectations conditioned on each feature, with the corresponding empirical expectations. However, the base model they use is a logistic regression model, which is a log-linear model, and hence would suffer from the same limitations as the linear SVM. There is also other work on combining SVMs with labeled features using transduction on unlabeled examples, that are soft-labeled using labeled features (Wu and Srihari, 2004), but we believe it will again suffer from the same limitations as the SVM on this domain.

In order to address the conjunctive and disjunctive errors simultaneously, we propose a new, but simple approach using propositional logic. We consider each labeled feature as a propositional variable, where true or false corresponds to whether the label of the feature is positive or negative respectively. Given a docket entry, we first extract its sentences, and for each sentence, we extract its labeled features, if present. Then, we construct a sentence-level formula formed by the conjunction of the variables representing the labeled features. The final classifier is a disjunction of the formulas of all sentences in the docket entry. Formally, the propositional logic based classifier can be expressed as follows:

$$C(D) = \vee_{i=1}^{N(D)} (\wedge_{j=1}^{M_i} L(f_{ij})) \tag{1}$$

where D is the docket entry, $N(D)$ is its number of sentences, M_i is the number of labeled features in the i^{th} sentence, f_{ij} is the j^{th} labeled feature in the i^{th} sentence and $L()$ is a mapping from a feature to its label, and $C(D)$ is the classification function where 'true' implies the docket entry contains an OSJ event.

The propositional logic model is designed to address the within-sentence conjunctive errors and without-sentence disjunctive errors simultaneously. Clearly, the within-sentence conjunctive behavior of the labeled features is captured by applying logical conjunctions to the labeled features within a sentence. Similarly, the disjunctive behavior of sentences is captured by applying disjunctions to the sentence-level clauses. This model requires no training, but for reasons of fairness in comparison, at testing time, we used only those human features (and their labels) that exist in the training set in each cross-validation run. The performance of this new approach, listed in table 5 as entry 7, is slightly better than the best performing SVM in entry 6. The difference in performance in this case is statistically significant, as measured by a paired, 2-tailed t-test at 95% confidence level (p-value = 0.007).

Although the improvement for this model is statistically significant, it does not entirely match our

#	Model	Recall (%)	Precision (%)	F1 (%)
1	Linear SVM with uni/bigrams only	75.19	84.21	79.44
2	Linear SVM with uni/bigrams only FS100	82.47	83.69	83.08*
3	Linear SVM with HF only	84.68	88.97	86.77*
4	Decision Tree with HF only	85.22	89.38	87.25
5	Quadratic SVM with HF only	84.14	90.98	87.43
6	Linear SVM with HF sentNgrams	84.63	93.37	88.78*
7	Propositional Logic with HF and their labels	**85.71**	**93.45**	**89.67***

Table 5: Results for 'Order re: Summary Judgment': FS100 indicates that only top 100 features were selected using Information Gain metric; HF stands for human built features, sentNgrams refers to the case where all the human-built features in a given sentence were merged to form an n-gram feature. A '*' next to F1 value indicates statistically significant result compared to its closest lower value, measured using a paired 2-tailed T-test, at 95% confidence level. The highest numbers in each column are highlighted using boldface.

expectations. Our data analysis showed a variety of errors caused mostly due to the following issues:

- *Imperfect sentence boundary detection:* since the propositional logic model considers sentences as strong conjunctions, it is more sensitive to errors in sentence boundary detection than SVMs. Any errors would cause the model to form conjunctions with features in neighboring sentences and deliver an incorrect labeling.

- *Incomplete feature set:* Some errors are caused because the feature set is not complete. For example, negative example 4 in table 2 is tagged as positive by the new model. This error could have been avoided if the word 'shorten' had been identified as a negative feature.

- *Relevant but bipolar features:* Although our model assumes that the selected features exhibit binary nature, this may not always be true. For example the word *allow* is sometimes used as a synonym for 'grant' which is a positive feature, but other times, as in negative example 8 in table 2, it exhibits negative polarity. Hence it is not always possible to encode all relevant features into the logic based model.

- *Limitations in expressiveness:* Some natural language sentences such as negative example 5 in table 2 are simply beyond the scope of the conjunctive and disjunctive formulations.

4 Discussion and Conclusions

Clearly, there is a significant amount of work to be done to further improve the performance of the propositional logic based classifier. One obvious line of work is towards better feature selection in this domain. One plausible technique would be to use shallow natural language processing techniques to extract the operative verbs acting on the phrase "summary judgment", and use them as the pruned feature set.

Another potential direction would be to extend the SVM-based system to model disjunctive behavior of sentences.[11] One way to accomplish this would be to classify each sentence individually and then to combine the outcomes using a disjunction. But for this to be implemented, we would also need labels at the sentence level during training time. One could procure these labels from annotators, but as an alternative, one could learn the sentence-level labels in an unsupervised fashion using a latent variable at the sentence level, but a supervised model at the docket-entry level. Such models may also be appropriate for traditional document classification where each document could be multi-labeled, and it is something we would like attempt in the future.

In addition, instead of manually constructing the logic based system, one could also automatically learn the rules by using ideas from earlier work on ILP (Muggleton, 1997), FOIL (Quinlan and Cameron-Jones, 1993), etc.

[11]Recall that the heuristics we presented for SVMs only address the conjunctive errors.

To summarize, we believe it is remarkable that a simple logic-based classifier could outperform an SVM that is already boosted by hand picked features and heuristics such as sentence level n-grams. This work clearly exposes some of the limitations of the state-of-the-art models in capturing the intricacies of natural language, and suggests that there is more work to be done in improving the performance of text based classifiers in specialized domains. As such, we hope our work motivates other researchers towards building better classifiers for this and other related problems.

Acknowledgments

The authors would like to thank Stanford university for financial support.

References

Chidanand Apté, Fred Damerau, and Sholom M. Weiss. 1994. Automated learning of decision rules for text categorization. *ACM Trans. Inf. Syst.*, 12(3):233–251.

R. Bekkerman, R. El-Yaniv, N. Tishby, and Y. Winter. 2003. Distributional word clusters vs. words for text categorization. *Journal of Machine Learning Research*.

Gregory Druck, Gideon Mann, and Andrew McCallum. 2008. Learning from labeled features using generalized expectation criteria. In *Proceedings of ACM Special Interest Group on Information Retreival, (SIGIR)*.

Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. 1998. Inductive learning algorithms and representations for text categorization. In *CIKM '98: Proceedings of the seventh international conference on Information and knowledge management*, pages 148–155, New York, NY, USA. ACM.

T. Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *ECML-98: 10th European Conference on Machine Learning*.

D.D. Lewis and M. Ringuette. 1994. Comparison of two learning algorithms for text categorization. In *Third Annual Symposium on Document Analysis and Information Retrieval*.

A. McCallum and K. Nigam. 1998. A comparison of event models for Naïve Bayes text classification . In *AAAI-98 Workshop on Learning for Text Categorization*.

Stephen Muggleton. 1997. *Inductive Logic Programming: 6th International Workshop: Seleted Papers*. Springer.

J.R. Quinlan and R.M. Cameron-Jones. 1993. Foil: a mid-term report. In *Proceedings of European Conference on Machine Learning*.

Hema Raghavan, Omid Madani, and Rosie Jones. 2006. Active learning with feedback on features and instances. *J. Mach. Learn. Res.*, 7:1655–1686.

Xiaoyun Wu and Rohini Srihari. 2004. Incorporating prior knowledge with weighted margin support vector machines. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 326–333, New York, NY, USA. ACM.

Yiming Yang and Xin Liu. 1999. A re-examination of text categorization methods. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49, New York, NY, USA. ACM.

Yiming Yang and Jan O. Pedersen. 1997. A comparative study on feature selection in text categorization. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 412–420, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Tong Zhang and Frank J. Oles. 2001. Text categorization based on regularized linear classification methods. *Information Retrieval*, 4(1):5–31.

A Discriminative Candidate Generator for String Transformations

Naoaki Okazaki[†]
okazaki@is.s.u-tokyo.ac.jp

Yoshimasa Tsuruoka[‡]
yoshimasa.tsuruoka@manchester.ac.uk

Sophia Ananiadou[‡]
sophia.ananiadou@manchester.ac.uk

Jun'ichi Tsujii[†‡]
tsujii@is.s.u-tokyo.ac.jp

[†]Graduate School of Information Science and Technology
University of Tokyo
7-3-1 Hongo, Bunkyo-ku
Tokyo 113-8656, Japan

[‡]School of Computer Science,
University of Manchester
National Centre for Text Mining (NaCTeM)
Manchester Interdisciplinary Biocentre
131 Princess Street, Manchester M1 7DN, UK

Abstract

String transformation, which maps a source string s into its desirable form t^*, is related to various applications including stemming, lemmatization, and spelling correction. The essential and important step for string transformation is to generate candidates to which the given string s is likely to be transformed. This paper presents a discriminative approach for generating candidate strings. We use substring substitution rules as features and score them using an L_1-regularized logistic regression model. We also propose a procedure to generate negative instances that affect the decision boundary of the model. The advantage of this approach is that candidate strings can be enumerated by an efficient algorithm because the processes of string transformation are tractable in the model. We demonstrate the remarkable performance of the proposed method in normalizing inflected words and spelling variations.

1 Introduction

String transformation maps a source string s into its destination string t^*. In the broad sense, string transformation can include labeling tasks such as part-of-speech tagging and shallow parsing (Brill, 1995). However, this study addresses string transformation in its narrow sense, in which a part of a source string is rewritten with a substring. Typical applications of this task include stemming, lemmatization, spelling correction (Brill and Moore, 2000; Wilbur et al., 2006; Carlson and Fette, 2007), OCR error correction (Kolak and Resnik, 2002), approximate string matching (Navarro, 2001), and duplicate record detection (Bilenko and Mooney, 2003).

Recent studies have formalized the task in the discriminative framework (Ahmad and Kondrak, 2005; Li et al., 2006; Chen et al., 2007),

$$t^* = \underset{t \in \mathrm{gen}(s)}{\mathrm{argmax}}\, P(t|s). \qquad (1)$$

Here, the *candidate generator* $\mathrm{gen}(s)$ enumerates candidates of destination (correct) strings, and the *scorer* $P(t|s)$ denotes the conditional probability of the string t for the given s. The scorer was modeled by a noisy-channel model (Shannon, 1948; Brill and Moore, 2000; Ahmad and Kondrak, 2005) and maximum entropy framework (Berger et al., 1996; Li et al., 2006; Chen et al., 2007).

The candidate generator $\mathrm{gen}(s)$ also affects the accuracy of the string transformation. Previous studies of spelling correction mostly defined $\mathrm{gen}(s)$,

$$\mathrm{gen}(s) = \{t \mid \mathrm{dist}(s,t) < \delta\}. \qquad (2)$$

Here, the function $\mathrm{dist}(s,t)$ denotes the weighted Levenshtein distance (Levenshtein, 1966) between strings s and t. Furthermore, the threshold δ requires the distance between the source string s and a candidate string t to be less than δ.

The choice of $\mathrm{dist}(s,t)$ and δ involves a tradeoff between the precision, recall, and training/tagging speed of the scorer. A less restrictive design of these factors broadens the search space, but it also increases the number of confusing candidates, amount of feature space, and computational cost for the scorer. Moreover, the choice is highly dependent on the target task. It might be sufficient for a spelling correction program to gather candidates from known words, but a stemmer must handle unseen words appropriately. The number of candidates can be huge when we consider transformations from and to unseen strings.

This paper addresses these challenges by exploring the discriminative training of candidate generators. More specifically, we build a binary classifier that, when given a source string s, decides whether a candidate t should be included in the candidate set or not. This approach appears straightforward, but it must resolve two practical issues. First, the task of the classifier is not only to make a binary decision for the two strings s and t, but also to enumerate a set of positive strings for the string s,

$$\mathrm{gen}(s) = \{t \mid \mathrm{predict}(s,t) = 1\}. \tag{3}$$

In other words, an efficient algorithm is necessary to find a set of strings with which the classifier $\mathrm{predict}(s,t)$ yields positive labels for the string s.

Another issue arises when we prepare a training set. A discriminative model requires a training set in which each instance (pair of strings) is annotated with a positive or negative label. Even though some existing resources (e.g., inflection table and query log) are available for positive instances, such resources rarely contain negative instances. Therefore, we must generate negative instances that are effective for discriminative training.

To address the first issue, we design features that express transformations from a source string s to its destination string t. Feature selection and weighting are performed using an L_1-regularized logistic regression model, which can find a sparse solution to the classification model. We also present an algorithm that utilizes the feature weights to enumerate candidates of destination strings efficiently. We deal with the second issue by generating negative instances from unlabeled instances. We describe a procedure to choose negative instances that affect the decision boundary of the classifier.

This paper is organized as follows. Section 2 formalizes the task of the candidate generator as a binary classification modeled by logistic regression. Features for the classifier are designed using the rules of substring substitution. Therefore, we can obtain, efficiently, candidates of destination strings and negative instances for training. Section 3 reports the remarkable performance of the proposed method in various applications including lemmatization, spelling normalization, and noun derivation. We briefly review previous work in Section 4, and conclude this paper in Section 5.

2 Candidate generator

2.1 Candidate classification model

In this section, we first introduce a binary classifier that yields a label $y \in \{0,1\}$ indicating whether a candidate t should be included in the candidate set (1) or not (0), given a source string s. We express the conditional probability $P(y|s,t)$ using a logistic regression model,

$$P(1|s,t) = \frac{1}{1 + \exp\left(-\mathbf{\Lambda}^T \boldsymbol{F}(s,t)\right)}, \tag{4}$$

$$P(0|s,t) = 1 - P(1|s,t). \tag{5}$$

In these equations, $\boldsymbol{F} = \{f_1, ..., f_K\}$ denotes a vector of the Boolean feature functions; K is the number of feature functions; and $\mathbf{\Lambda} = \{\lambda_1, ..., \lambda_K\}$ presents a weight vector of the feature functions.

We obtain the following decision rule to choose the most probable label y^* for a given pair $\langle s, t\rangle$,

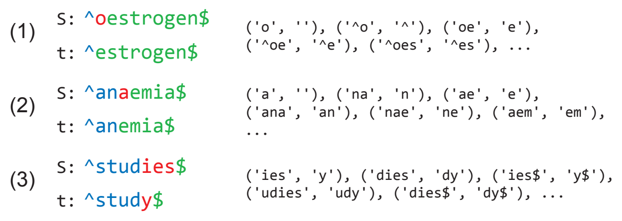
$$y^* = \operatorname*{argmax}_{y \in \{0,1\}} P(y|s,t) = \begin{cases} 1 & \left(\mathbf{\Lambda}^T \boldsymbol{F}(s,t) > 0\right) \\ 0 & (\text{otherwise}) \end{cases}. \tag{6}$$

Finally, given a source string s, the generator function $\mathrm{gen}(s)$ is defined to collect all strings to which the classifier assigns positive labels:

$$\begin{aligned} \mathrm{gen}(s) &= \{t \mid P(1|s,t) > P(0|s,t)\} \\ &= \{t \mid \mathbf{\Lambda}^T \boldsymbol{F}(s,t) > 0\}. \end{aligned} \tag{7}$$

2.2 Substitution rules as features

The binary classifier can include any arbitrary feature. This is exemplified by the Levenshtein distance and distributional similarity (Lee, 1999) between two strings s and t. These features can improve the classification accuracy, but it is unrealistic to compute these features for every possible string, as in equation 7. For that reason, we specifically examine *substitution rules*, with which the process



```
(1)  S: ^oestrogen$    ('o', ''), ('^o', '^'), ('oe', 'e'),
     t: ^estrogen$     ('^oe', '^e'), ('^oes', '^es'), ...
(2)  S: ^anaemia$      ('a', ''), ('na', 'n'), ('ae', 'e'),
     t: ^anemia$       ('ana', 'an'), ('nae', 'ne'), ('aem', 'em'),
                       ...
(3)  S: ^studies$      ('ies', 'y'), ('dies', 'dy'), ('ies$', 'y$'),
     t: ^study$        ('udies', 'udy'), ('dies$', 'dy$'), ...
```

Figure 1: Generating substitution rules.

of transforming a source string s into its destination form t is tractable.

In this study, we assume that every string has a prefix '`^`' and postfix '`$`', which indicate the head and tail of a string. A substitution rule $r = (\alpha, \beta)$ replaces every occurrence of the substring α in a source string into the substring β. Assuming that a string s can be transformed into another string t with a single substitution operation, substitution rules express the different portion between strings s and t.

Equation 8 defines a binary feature function with a substitution rule between two strings s and t,

$$f_k(s,t) = \begin{cases} 1 & \text{(rule } r_k \text{ can convert } s \text{ into } t\text{)} \\ 0 & \text{(otherwise)} \end{cases}. \tag{8}$$

We allow multiple substitution rules for a given pair of strings. For instance, substitution rules ('`a`', ''), ('`na`', '`n`'), ('`ae`', '`e`'), ('`nae`', '`ne`'), etc. form feature functions that yield 1 for strings s = '`^anaemia$`' and t = '`^anemia$`'. Equation 6 produces a decision based on the sum of feature weights, or scores of substitution rules, representing the different portions between s and t.

Substitution rules for the given two strings s and t are obtained as follows. Let l denote the longest common prefix between strings s and t, and r the longest common postfix. We define c_s as the substring in s that is not covered by the longest common prefix l and postfix r, and define c_t for t analogously. In other words, strings s and t are divided into three regions, lc_sr and lc_tr, respectively. For strings s = '`^anaemia$`' and t = '`^anemia$`' in Figure 1 (2), we obtain c_s = '`a`' and c_t = '' because l = '`^an`' and r = '`emia$`'.

Because substrings c_s and c_t express different portions between strings s and t, we obtain the *minimum substitution rule* (c_s, c_t), which can convert the string s into t by replacing substrings c_s in s with c_t; the minimum substitution rule for the same example is ('`a`', ''). However, replacing letters '`a`' in '`^anaemia$`' into empty letters does not produce the correct string '`^anemia$`' but '`^nemi$`'. Furthermore, the rule might be inappropriate for expressing string transformation because it always removes the letter '`a`' from every string.

Therefore, we also obtain *expanded substitution rules*, which insert postfixes of l to the head of minimum substitution rules, and/or append prefixes of r to the rules. For example, we find an expanded substitution rule ('`na`', '`n`'), by inserting a postfix of l = '`^an`' to the head of the minimum substitution rule ('`a`', ''); similarly, we obtain an expanded substitution rule ('`ae`', '`e`'), by appending a prefix of r = '`emia$`' to the tail of the rule ('`a`', '').

Figure 1 displays examples of substitution rules (the right side) for three pairs of strings (the left side). Letters in blue, green, and red respectively represent the longest common prefixes, longest common postfixes, and different portions. In this study, we expand substitution rules such that the number of letters in rules is does not pass a threshold θ[1].

2.3 Parameter estimation

Given a training set that consists of N instances, $\mathcal{D} = \left((s^{(1)}, t^{(1)}, y^{(1)}), ..., (s^{(N)}, t^{(N)}, y^{(N)})\right)$, we optimize the feature weights in the logistic regression model by maximizing the log-likelihood of the conditional probability distribution,

$$\mathcal{L}_{\mathbf{\Lambda}} = \sum_{i=1}^{N} \log P(y^{(i)}|s^{(i)}, t^{(i)}). \tag{9}$$

The partial derivative of the log-likelihood with respect to a feature weight λ_k is given as equation 10,

$$\frac{\partial \mathcal{L}_{\mathbf{\Lambda}}}{\partial \lambda_k} = \sum_{i=1}^{N} \left\{ y^{(i)} - P(1|s^{(i)}, t^{(i)}) \right\} f_k(s^{(i)}, t^{(i)}). \tag{10}$$

The maximum likelihood estimation (MLE) is known to suffer from overfitting the training set. The

[1]The number of letters for a substitution rule $r = (\alpha, \beta)$ is defined as the sum of the quantities of letters in α and β, i.e., $|\alpha| + |\beta|$. We determined the threshold $\theta = 12$ experimentally.

common approach for addressing this issue is to use the maximum a posteriori (MAP) estimation, introducing a regularization term of the feature weights $\mathbf{\Lambda}$, i.e., a penalty on large feature weights. In addition, the generation algorithm of substitution rules might produce inappropriate rules that transform a string incorrectly, or overly specific rules that are used scarcely. Removing unnecessary substitution rules not only speeds up the classifier but also the algorithm for candidate generation, as presented in Section 2.4.

In recent years, L_1 regularization has received increasing attention because it produces a sparse solution of feature weights in which numerous feature weights are zero (Tibshirani, 1996; Ng, 2004). Therefore, we regularize the log-likelihood with the L_1 norm of the weight vector $\mathbf{\Lambda}$ and define the final form the objective function to be minimized as

$$E_{\mathbf{\Lambda}} = -\mathcal{L}_{\mathbf{\Lambda}} + \frac{|\mathbf{\Lambda}|}{\sigma}. \tag{11}$$

Here, σ is a parameter to control the effect of L_1 regularization; the smaller the value we set to σ, the more features the MAP estimation assigns zero weights to: it removes a number of features from the model. Equation 11 is minimized using the Orthant-Wise Limited-memory Quasi-Newton (OW-LQN) method (Andrew and Gao, 2007) because the second term of equation 11 is not differentiable at $\lambda_k = 0$.

2.4 Candidate generation

The advantage of our feature design is that we can enumerate strings to which the classifier is likely to assign positive labels. We start by observing the necessary condition for t in equation 7,

$$\mathbf{\Lambda}^T \boldsymbol{F}(s,t) > 0 \Rightarrow \exists k : f_k(s,t) = 1 \land \lambda_k > 0. \tag{12}$$

The classifier might assign a positive label to strings s and t when at least one feature function whose weight is positive can transform s to t.

Let R^+ be a set of substitution rules to which MAP estimation has assigned positive feature weights. Because each feature corresponds to a substitution rule, we can obtain $\mathrm{gen}(s)$ for a given string s by application of every substitution rule $r \in R^+$,

$$\mathrm{gen}(s) = \{r(s) \mid r \in R^+ \land \mathbf{\Lambda}^T \boldsymbol{F}(s, r(s)) > 0\}. \tag{13}$$

```
Input: $s = (s_1, ..., s_l)$: an input string $s$ (series of letters)
Input: $D$: a trie dictionary containing positive features
Output: $T$: $\mathrm{gen}(s)$
1  $T = \{\}$;
2  $U = \{\}$;
3  foreach $i \in (1, ..., |s|)$ do
4      $F \leftarrow D$.prefix_search($s, i$);
5      foreach $f \in F$ do
6          if $f \notin U$ then
7              $t \leftarrow f$.apply($s$);
8              if classify($s, t$) $= 1$ then
9                  add $t$ to $T$;
10             end
11             add $f$ to $U$;
12         end
13     end
14 end
15 return $T$;
```

Algorithm 1: A pseudo-code for $\mathrm{gen}(s)$.

Here, $r(s)$ presents the string to which the substitution rule r transforms the source string s. We can compute $\mathrm{gen}(s)$ with a small computational cost if the MAP estimation with L_1 regularization reduces the number of active features.

Algorithm 1 represents a pseudo-code for obtaining $\mathrm{gen}(s)$. To search for positive substitution rules efficiently, the code stores a set of rules in a trie structure. In line 4, the code obtains a set of positive substitution rules F that can rewrite substrings starting at offset #i in the source string s. For each rule $f \in F$, we obtain a candidate string t by application of the substitution rule f to the source string s (line 7). The candidate string t is qualified to be included in $\mathrm{gen}(s)$ when the classifier assigns a positive label to strings s and t (lines 8 and 9). Lines 6 and 11 prevent the algorithm from repeating evaluation of the same substitution rule.

2.5 Generating negative instances

The parameter estimation requires a training set $\mathcal{D}$ in which each instance (pair of strings) is annotated with a positive or negative label. Negative instances (counter examples) are essential for penalizing inappropriate substitution rules, e.g. (‘a’, ‘’). Even though some existing resources (e.g. verb inflection table) are available for positive instances, such resources rarely contain negative instances.

A common approach for handling this situation is to assume that every pair of strings in a resource

```
Input: $\mathcal{D}^+ = [(s_1, t_1), ..., (s_l, t_l)]$: positive instances
Input: $V$: a suffix array of all strings (vocabulary)
Output: $\mathcal{D}^-$: negative instances
Output: $R$: substitution rules (features)
1  $\mathcal{D}^- = []$;
2  $R = \{\}$;
3  foreach $d \in \mathcal{D}^+$ do
4      foreach $r \in$ features($d$) do
5          add $r$ to $R$;
6      end
7  end
8  foreach $r \in R$ do
9      $S \leftarrow V$.search($r.src$);
10     foreach $s \in S$ do
11         $t \leftarrow r$.apply($s$);
12         if $(s, t) \notin \mathcal{D}^+$ then
13             if $t \in V$ then
14                 append $(s, t)$ to $\mathcal{D}^-$;
15             end
16         end
17     end
18 end
19 return $\mathcal{D}^-$, $R$;
```

Algorithm 2: Generating negative instances.

is a negative instance; however, negative instances amount to ca. $V(V-1)/2$, where V represents the total number of strings. Moreover, substitution rules expressing negative instances are innumerable and sparse because the different portions are peculiar to individual negative instances. For instance, the minimum substitution rule for unrelated words *anaemia* and *around* is (‘naemia’, ‘round’), but the rule cannot be too specific to generalize the conditions for other negative instances.

In this study, we generate negative instances so that they can penalize inappropriate rules and settle the decision boundary of the classifier. This strategy is summarized as follows. We consider every pair of strings as candidates for negative instances. We obtain substitution rules for the pair using the same algorithm as that described in Section 2.2 if a string pair is not included in the dictionary (i.e., not in positive instances). The pair is used as a negative instance only when any substitution rule generated from the pair also exists in the substitution rules generated from positive instances.

Algorithm 2 presents the pseudo-code that implements the strategy for generating negative instances efficiently. First, we presume that we have positive instances $\mathcal{D}^+ = [(s_1, t_1), ..., (s_l, t_l)]$ and unlabeled strings V. For example, positive instance $\mathcal{D}^+$ represent orthographic variants, and unlabeled strings V include all possible words (vocabulary). We insert the vocabulary into a suffix array, which is used to locate every occurrence of substrings in V.

The algorithm first generates substitution rules R only from positive instances $\mathcal{D}^+$ (lines 3 to 7). For each substitution rule $r \in R$, we enumerate known strings S that contain the source substring $r.src$ (line 9). We apply the substitution rule to each string $s \in S$ and obtain its destination string t (line 11). If the pair of strings $\langle s, t \rangle$ is not included in D^+ (line 12), and if the destination string t is known (line 13), the substitution rule r might associate incorrect strings s and t, which do not exist in D^+. Therefore, we insert the pair to the negative set D^- (line 14).

Table	Description	# Entries
LRSPL	Spelling variants	90,323
LRNOM	Nominalizations (derivations)	14,029
LRAGR	Agreement and inflection	910,854
LRWD	Word index (vocabulary)	850,236

Table 1: Excerpt of tables in the SPECIALIST Lexicon.

Data set	# +	# -	# Rules
Orthography	15,830	33,296	11,098
Derivation	12,988	85,928	5,688
Inflection	113,215	124,747	32,278

Table 2: Characteristics of datasets.

3 Evaluation

3.1 Experiments

We evaluated the candidate generator using three different tasks: normalization of orthographic variants, noun derivation, and lemmatization. The datasets for these tasks were obtained from the *UMLS SPECIALIST Lexicon*[2], a large lexicon that includes both commonly occurring English words and biomedical vocabulary. Table 1 displays the list of tables in the SPECIALIST Lexicon that were used in our experiments. We prepared three datasets, *Orthography*, *Derivation*, and *Inflection*.

The Orthography dataset includes spelling variants (e.g., *color* and *colour*) in the LRSPL table. We

[2]UMLS SPECIALIST Lexicon: http://specialist.nlm.nih.gov/

chose entries as positive instances in which spelling variants are caused by (case-insensitive) alphanumeric changes[3]. The Derivation dataset was built directly from the LRNOM table, which includes noun derivations such as *abandon* $\rightarrow$ *abandonment*. The LRAGR table includes base forms and their inflectional variants of nouns (singular and plural forms), verbs (infinitive, third singular, past, past participle forms, etc), and adjectives/adverbs (positive, comparative, and superlative forms). For the Inflection dataset, we extracted the entries in which inflectional forms differ from their base forms[4], e.g., *study* $\rightarrow$ *studies*.

For each dataset, we applied the algorithm described in Section 2.5 to generate substitution rules and negative instances. Table 2 shows the number of positive instances (# +), negative instances (# -), and substitution rules (# Rules). We evaluated the performance of the proposed method in two different goals of the tasks: classification (Section 3.2) and normalization (Section 3.3).

3.2 Experiment 1: Candidate classification

In this experiment, we measured the performance of the classification task in which pairs of strings were assigned with positive or negative labels. We trained and evaluated the proposed method by performing ten-fold cross validation on each dataset[5]. Eight baseline systems were prepared for comparison: Levenshtein distance (LD), normalized Levenshtein distance (NLD), Dice coefficient on letter bigrams (DICE) (Adamson and Boreham, 1974), Longest Common Substring Ratio (LCSR) (Melamed, 1999), Longest Common Prefix Ratio (PREFIX) (Kondrak, 2005), Porter's stemmer (Porter, 1980), Morpha (Minnen et al., 2001), and CST's lemmatiser (Dalianis and Jongejan, 2006)[6].

The five systems LD, NLD, DICE, LCSR, and PREFIX employ corresponding metrics of string distance or similarity. Each system assigns a positive label to a given pair of strings $\langle$s, t$\rangle$ if the distance/similarity of strings s and t is smaller/larger than the threshold δ (refer to equation 2 for distance metrics). The threshold of each system was chosen so that the system achieves the best F1 score.

The remaining three systems assign a positive label only if the system transforms the strings s and t into the identical string. For example, a pair of two words *studies* and *study* is classified as positive by Porter's stemmer, which yields the identical stem *studi* for these words. We trained CST's lemmatiser for each dataset to obtain flex patterns that are used for normalizing word inflections.

To examine the performance of the L_1-regularized logistic regression as a discriminative model, we also built two classifiers based on the Support Vector Machine (SVM). These SVM classifiers were implemented by the SVM^{perf}[7] on a linear kernel[8]. An SVM classifier employs the same feature set (substitution rules) as the proposed method so that we can directly compare the L_1-regularized logistic regression and the linear-kernel SVM. Another SVM classifier incorporates the five string metrics; this system can be considered as our reproduction of the discriminative string similarity proposed by Bergsma and Kondrak (2007).

Table 3 reports the precision (P), recall (R), and F1 score (F1) based on the number of correct decisions for positive instances. The proposed method outperformed the baseline systems, achieving 0.919, 0.888, and 0.984 of F1 scores, respectively. Porter's stemmer worked on the Inflection set, but not on the Orthography set, which is beyond the scope of the stemming algorithms. CST's lemmatizer suffered from low recall on the Inflection set because it removed suffixes of base forms, e.g., (*cloning*, *clone*) $\rightarrow$ (*clone*, *clo*). Morpha and CST's lemma-

[3]LRSPL table includes trivial spelling variants that can be handled using simple character/string operations. For example, the table contains spelling variants related to case sensitivity (e.g., *deg* and *Deg*) and symbols (e.g., *Feb* and *Feb.*).

[4]LRAGR table also provides agreement information even when word forms do not change. For example, the table contains an entry indicating that the first-singular present form of the verb *study* is *study*, which might be readily apparent to English speakers.

[5]We determined the regularization parameter $\sigma = 5$ experimentally. Refer to Figure 2 for the performance change.

[6]We used CST's lemmatiser version 2.13: `http://www.cst.dk/online/lemmatiser/uk/index.html`

[7]SVM for Multivariate Performance Measures (SVM^{perf}): `http://svmlight.joachims.org/svm_perf.html`

[8]We determined the parameter $C = 500$ experimentally; it controls the tradeoff between training error and margin.

System	Orthography			Derivation			Inflection		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Levenshtein distance ($\delta = 1$)	.319	.871	.467	.004	.006	.005	.484	.679	.565
Levenshtein distance	.323	.999	.488	.131	1.00	.232	.479	.988	.646
Normalized Levenshtein distance	.441	.847	.580	.133	.990	.235	.598	.770	.673
Dice coefficient (letter bigram)	.401	.918	.558	.137	.984	.240	.476	1.00	.645
LCSR	.322	1.00	.487	.156	.841	.263	.476	1.00	.645
PREFIX	.418	.927	.576	.140	.943	.244	.476	1.00	.645
Porter stemmer (Porter, 1980)	.084	.074	.079	.197	.846	.320	.926	.839	.881
Morpha (Minnen et al., 2001)	.009	.007	.008	.012	.022	.016	.979	.836	.902
CST's lemmatiser (Dalianis et al. 2006)	.119	.008	.016	.383	.682	.491	.821	.176	.290
Proposed method	.941	.898	.919	**.896**	.880	.888	**.985**	.986	**.984**
Substitution rules trained with SVM	.943	.890	.916	.894	**.886**	**.890**	.980	**.987**	.983
+ LD, NLD, DICE, LCSR, PREFIX	**.946**	**.906**	**.926**	.894	**.886**	**.890**	.980	**.987**	.983

Table 3: Performance of candidate classification

Rank	Src	Dst	Weight	Examples
1	uss	us	9.81	focussing
2	aev	ev	9.56	mediaeval
3	aen	en	9.53	ozaena
4	iae$	ae$	9.44	gadoviae
5	nni	ni	9.16	prorennin
6	nne	ne	8.84	connexus
7	our	or	8.54	colour
8	aea	ea	8.31	paean
9	aeu	eu	8.22	stomodaeum
10	ooll	ool	7.79	woollen

Table 4: Feature weights for the Orthography set

tizer were not designed for orthographic variants and noun derivations.

Levenshtein distance ($\delta = 1$) did not work for the Derivation set because noun derivations often append two or more letters (e.g., *happy* $\rightarrow$ *happiness*). No string similarity/distance metrics yielded satisfactory results. Some metrics obtained the best F1 scores with extreme thresholds only to classify every instance as positive. These results imply the difficulty of the string metrics for the tasks.

The L_1-regularized logistic regression was comparable to the SVM with linear kernel in this experiment. However, the presented model presents the advantage that it can reduce the number of active features (features with non-zero weights assigned); the L_1 regularization can remove 74%, 48%, and 82% of substitution rules in each dataset. The performance improvements by incorporating string metrics as features were very subtle (less than 0.7%). What is worse, the distance/similarity metrics do not specifically derive destination strings to which the classifier is likely to assign positive labels. Therefore, we can no longer use the efficient algorithm as a candidate generator (in Section 2.4) with these features.

Table 4 demonstrates the ability of our approach to obtain effective features; the table shows the top 10 features with high weights assigned for the Orthography data. An interesting aspect of the proposed method is that the process of the orthographic variants is interpretable through the feature weights.

Figure 2 shows plots of the F1 scores (y-axis) for the Inflection data when we change the number of active features (x-axis) by controlling the regularization parameter σ from 0.001 to 100. The larger the value we set for σ, the better the classifier performs, generally, with more active features. In extreme cases, the number of active features drops to 97 with $\sigma = 0.01$; nonetheless, the classifier still achieves 0.961 of the F1 score. The result suggests that a small set of substitution rules can accommodate most cases of inflectional variations.

3.3 Experiment 2: String transformation

The second experiment examined the performance of the string normalization tasks formalized in equation 1. In this task, a system was given a string s and was required to yield either its transformed form t^* ($s \neq t^*$) or the string s itself when the transformation is unnecessary for s. The conditional probability distribution (scorer) in equation 1 was modeled

System	Orthography			Derivation			Inflection			XTAG morph 1.5		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Morpha	.078	.012	.021	.233	.016	.029	.435	.682	.531	.830	.587	.688
CST's lemmatiser	.135	.160	.146	.378	.732	.499	.367	.762	.495	.584	.589	.587
Proposed method	**.859**	**.823**	**.841**	**.979**	**.981**	**.980**	**.973**	**.979**	**.976**	**.837**	**.816**	**.827**

Table 5: Performance of string transformation

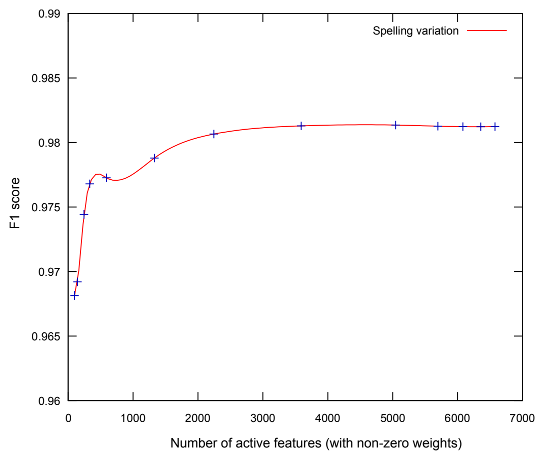


Figure 2: Number of active features and performance.

by the maximum entropy framework. Features for the maximum entropy model consist of: substitution rules between strings s and t, letter bigrams and trigrams in s, and letter bigrams and trigrams in t.

We prepared four datasets, *Orthography*, *Derivation*, *Inflection*, and *XTAG morphology*. Each dataset is a list of string pairs $\langle s, t\rangle$ that indicate the transformation of the string s into t. A source string s is identical to its destination string t when string s should not be changed. These instances correspond to the case where string s has already been lemmatized. For each string pair (s, t) in LRSPL[9], LRNOM, and LRAGR tables, we generated two instances $\langle s, t\rangle$ and $\langle t, t\rangle$. Consequently, a system is expected to leave the string t unchanged. We also used XTAG morphology[10] to perform a cross-domain evaluation of the lemmatizer trained on the Inflection dataset[11]. The entries in XTAG morphology that also appear in the Inflection dataset were 39,130 out of 317,322 (12.3 %). We evaluated the proposed method and CST's lemmatizer by performing ten-fold cross validation.

Table 5 reports the performance based on the number of correct transformations. The proposed method again outperformed the baseline systems with a wide margin. It is noteworthy that the proposed method can accommodate morphological inflections in the XTAG morphology corpus with no manual tuning or adaptation.

Although we introduced no assumptions about target tasks (e.g. a known vocabulary), the average number of positive substitution rules relevant to source strings was as small as 23.9 (in XTAG morphology data). Therefore, the candidate generator performed 23.9 substitution operations for a given string. It applied the decision rules (equation 7) 21.3 times, and generated 1.67 candidate strings per source string. The experimental results described herein demonstrated that the candidate generator was modeled successfully by the discriminative framework.

4 Related work

The task of string transformation has a long history in natural language processing and information retrieval. As described in Section 1, this task is related closely to various applications. Therefore, we specifically examine several prior studies that are relevant to this paper in terms of technical aspects.

Some researchers have reported the effectiveness of the discriminative framework of string similarity. MaCallum et al. (2005) proposed a method to train the costs of edit operations using Conditional Random Fields (CRFs). Bergsma and Kondrak (2007)

[9]We define that s precedes t in dictionary order.

[10]XTAG morphology database 1.5:
`ftp://ftp.cis.upenn.edu/pub/xtag/morph-1.5/morph-1.5.tar.gz`

[11]We found that XTAG morphology contains numerous incorrect comparative and superlative adjectives, e.g., *unpopular* → *unpopularer* → *unpopularest* and *refundable* → *refundabler* → *refundablest*. Therefore, we removed inflection entries for comparative and superlative adjectives from the dataset.

presented an alignment-based discriminative string similarity. They extracted features from substring pairs that are consistent to a character-based alignment of two strings. Aramaki et al. (2008) also used features that express the different segments of the two strings. However, these studies are not suited for a candidate generator because the processes of string transformations are intractable in their discriminative models.

Dalianis and Jongejan (2006) presented a lemmatiser based on suffix rules. Although they proposed a method to obtain suffix rules from a training data, the method did not use counter-examples (negatives) for reducing incorrect string transformations. Tsuruoka et al. (2008) proposed a scoring method for discovering a list of normalization rules for dictionary look-ups. However, their objective was to transform given strings, so that strings (e.g., *studies* and *study*) referring to the same concept in the dictionary are mapped into the same string (e.g., *stud*); in contrast, this study maps strings into their destination strings that were specified by the training data.

5 Conclusion

We have presented a discriminative approach for generating candidates for string transformation. Unlike conventional spelling-correction tasks, this study did not assume a fixed set of destination strings (e.g. correct words), but could even generate unseen candidate strings. We used an L_1-regularized logistic regression model with substring-substitution features so that candidate strings for a given string can be enumerated using the efficient algorithm. The results of experiments described herein showed remarkable improvements and usefulness of the proposed approach in three tasks: normalization of orthographic variants, noun derivation, and lemmatization.

The method presented in this paper allows only one region of change in string transformation. A natural extension of this study is to handle multiple regions of changes for morphologically rich languages (e.g. German) and to handle changes at the phrase/term level (e.g., "estrogen receptor" and "receptor of oestrogen"). Another direction would be to incorporate the methodologies for semi-supervised machine learning to accommodate situations in which positive instances and/or unlabeled strings are insufficient.

Acknowledgments

This work was partially supported by Grants-in-Aid for Scientific Research on Priority Areas (MEXT, Japan), and for Solution-Oriented Research for Science and Technology (JST, Japan).

References

George W. Adamson and Jillian Boreham. 1974. The use of an association measure based on character structure to identify semantically related pairs of words and document titles. *Information Storage and Retrieval*, 10(7-8):253–260.

Farooq Ahmad and Grzegorz Kondrak. 2005. Learning a spelling error model from search query logs. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT-EMNLP 2005)*, pages 955–962.

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L_1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning (ICML 2007)*, pages 33–40.

Eiji Aramaki, Takeshi Imai, Kengo Miyo, and Kazuhiko Ohe. 2008. Orthographic disambiguation incorporating transliterated probability. In *Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP 2008)*, pages 48–55.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Shane Bergsma and Grzegorz Kondrak. 2007. Alignment-based discriminative string similarity. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL 2007)*, pages 656–663.

Mikhail Bilenko and Raymond J. Mooney. 2003. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD 2003)*, pages 39–48.

Eric Brill and Robert C. Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on the Association for Computational Linguistics (ACL 2000)*, pages 286–293.

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

Andrew Carlson and Ian Fette. 2007. Memory-based context-sensitive spelling correction at web scale. In *Proceedings of the Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*, pages 166–171.

Qing Chen, Mu Li, and Ming Zhou. 2007. Improving query spelling correction using web search results. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, pages 181–189.

Hercules Dalianis and Bart Jongejan. 2006. Handcrafted versus machine-learned inflectional rules: The euroling-siteseeker stemmer and cst's lemmatiser. In *In Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2006)*, pages 663–666.

Okan Kolak and Philip Resnik. 2002. OCR error correction using a noisy channel model. In *Proceedings of the second international conference on Human Language Technology Research (HLT 2002)*, pages 257–262.

Grzegorz Kondrak. 2005. Cognates and word alignment in bitexts. In *Proceedings of the Tenth Machine Translation Summit (MT Summit X)*, pages 305–312.

Lillian Lee. 1999. Measures of distributional similarity. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL 1999)*, pages 25–32.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

Mu Li, Yang Zhang, Muhua Zhu, and Ming Zhou. 2006. Exploring distributional similarity based models for query spelling correction. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (Coling-ACL 2006)*, pages 1025–1032.

Andrew McCallum, Kedar Bellare, and Fernando Pereira. 2005. A conditional random field for discriminatively-trained finite-state string edit distance. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI 2005)*, pages 388–395.

I. Dan Melamed. 1999. Bitext maps and alignment via pattern recognition. *Computational Linguistics*, 25(1):107–130.

Guido Minnen, John Carroll, and Darren Pearce. 2001. Applied morphological processing of English. *Natural Language Engineering*, 7(3):207–223.

Gonzalo Navarro. 2001. A guided tour to approximate string matching. *ACM Computing Surveys (CSUR)*, 33(1):31–88.

Andrew Y. Ng. 2004. Feature selection, L_1 vs. L_2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning (ICML 2004)*, pages 78–85.

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

Claude E. Shannon. 1948. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423.

Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288.

Yoshimasa Tsuruoka, John McNaught, and Sophia Ananiadou. 2008. Normalizing biomedical terms by minimizing ambiguity and variability. *BMC Bioinformatics*, Suppl 3(9):S2.

W. John Wilbur, Won Kim, and Natalie Xie. 2006. Spelling correction in the PubMed search engine. *Information Retrieval*, 9(5):543–564.

Automatic induction of FrameNet lexical units

Marco Pennacchiotti$^{(\dagger)}$, **Diego De Cao**$^{(\ddagger)}$, **Roberto Basili**$^{(\ddagger)}$, **Danilo Croce**$^{(\ddagger)}$, **Michael Roth**$^{(\dagger)}$

(†) Computational Linguistics
Saarland University
Saarbrücken, Germany
{pennacchiotti,mroth}@coli.uni-sb.de

(‡) DISP
University of Roma Tor Vergata
Roma, Italy
{decao,basili,croce}@info.uniroma2.it

Abstract

Most attempts to integrate FrameNet in NLP systems have so far failed because of its limited coverage. In this paper, we investigate the applicability of distributional and WordNet-based models on the task of *lexical unit induction*, i.e. the expansion of FrameNet with new lexical units. Experimental results show that our distributional and WordNet-based models achieve good level of accuracy and coverage, especially when combined.

1 Introduction

Most inference-based NLP tasks require a large amount of semantic knowledge at the predicate-argument level. This type of knowledge allows to identify meaning-preserving transformations, such as active/passive, verb alternations and nominalizations, which are crucial in several linguistic inferences. Recently, the integration of NLP systems with manually-built resources at the predicate argument-level, such as FrameNet (Baker et al., 1998) and PropBank (Palmer et al., 2005) has received growing interest. For example, Shen and Lapata (2007) show the potential improvement that FrameNet can bring on the performance of a Question Answering (QA) system. Similarly, several other studies (e.g. (Bar-Haim et al., 2005; Garoufi, 2007)) indicate that frame semantics plays a central role in Recognizing Textual Entailment (RTE). Unfortunately, most attempts to integrate FrameNet or similar resources in QA and RTE systems have so far failed, as reviewed respectively in (Shen and Lapata, 2007) and (Burchardt and Frank, 2006). These studies indicate *limited coverage* as the main reason of insuccess. Indeed, the FrameNet database only contains 10,000 lexical units (LUs), far less than the 210,000 entries in WordNet 3.0. Also, frames are based on more complex information than word senses, so that their manual development is much more demanding (Burchardt et al., 2006; Subirats and Petruck, 2003).

Therefore, there is nowadays a pressing need to adopt learning approaches to extend the coverage of the FrameNet lexicon by automatically acquiring new LUs, a task we call **LU induction**, as recently proposed at SemEval-2007 (Baker et al., 2007). Unfortunately, research in this area is still somehow limited and fragmentary. The aim of our study is to pioneer in this field by proposing two unsupervised models for LU induction, one based on distributional techniques and one using WordNet as a support; and a combined model which mixes the two. The goal is to investigate to what extent distributional and WordNet-based models can be used to induce frame semantic knowledge in order to safely extend FrameNet, thus limiting the high costs of manual annotation.

In Section 2 we introduce the LU induction task and present related work. In Sections 3, 4 and 5 we present our distributional, WordNet-based and combined models. Then, in Section 6 we report experimental results and comparative evaluations. Finally, in Section 7 we draw final conclusions and outline future work.

2 Task Definition and Related Work

As defined in (Fillmore, 1985), a frame is a conceptual structure modeling a prototypical situation, evoked in texts through the occurrence of its lexical units. A *lexical unit* (LU) is a predicate that linguistically expresses the situation of the frame. Lexical units of the same frame share semantic arguments. For example the frame KILLING has lexical units such as *assassin, assassinate, blood-bath, fatal, murderer, kill, suicide* that share semantic arguments such as KILLER, INSTRUMENT, CAUSE, VICTIM. Building on this frame-semantic model, the Berkeley FrameNet project (Baker et al., 1998) has been developing a frame-semantic lexicon for the core vocabulary of English since 1997. The current FrameNet release contains 795 frames and about 10,000 LUs. Part of FrameNet is also a corpus of 135,000 annotated example sentences from the British National Corpus (BNC).

LU induction is a fairly new task. Formally, it can be defined as the task of assigning a generic lexical unit not yet present in the FrameNet database (hereafter called *unknown LU*) to the correct frame(s). As the number of frames is very large (about 800) the task is intuitively hard to solve. A further complexity regards multiple assignments. Lexical units are sometimes ambiguous and can then be mapped to more than one frame (for example the word *tea* could map both to FOOD and SOCIAL_EVENT). Also, even unambiguous words can be assigned to more than one frame – e.g. *child* maps to both KINSHIP and PEOPLE_BY_AGE.

LU induction is relevant to many NLP tasks, such as the semi-automatic creation of new FrameNets, and semantic role labelling. LU induction has been integrated at SemEval-2007 as part of the Frame Semantic Structure Extraction shared task (Baker et al., 2007), where systems are requested to assign the correct frame to a given LU, even when the LU is not yet present in FrameNet. Johansson and Nugues (2007) approach the task as a machine learning problem: a Support Vector Machine trained on existing LUs is applied to assign unknown LUs to the correct frame, using features derived from the WordNet hierarchy. Tested on the FrameNet gold standard, the method achieves an accuracy of 0.78, at the cost of a low coverage of 31% (i.e. many LUs are not assigned). Johansson and Nugues (2007) also experiment with a simple model based on standard WordNet similarity measures (Pedersen et al., 2004), achieving lower performance. Burchardt and colleagues (2005) present Detour, a rule-based system using words in a WordNet relation with the unknown LU to find the correct frame. The system achieves an accuracy of 0.39 and a coverage of 87%. Unfortunately this algorithm requires the LU to be previously disambiguated, either by hand or using contextual information.

In a departure from previous work, our first model leverages distributional properties to induce LUs, instead of relying on pre-existing lexical resources as WordNet. This guarantees two main advantages. First, it can predict a frame for any unknown LU, while WordNet based approaches can be applied only to words having a WordNet entry. Second, it allows to induce LUs in languages for which WordNet is not available or has limited coverage. Our second WordNet-based model uses sense information to characterize the frame membership for unknown LU, by adopting a semantic similarity measure which is sensitive to *all* the known LUs of a frame.

3 Distributional model

The basic idea behind the distributional approach is to induce new LUs by modelling existing frames and unknown LUs in a semantic space, where they are represented as distributional co-occurrence vectors computed over a corpus.

Semantic spaces are widely used in NLP for representing the meaning of words or other lexical entities. They have been successfully applied in several tasks, such as information retrieval (Salton et al., 1975) and harvesting thesauri (Lin, 1998). The intuition is that the meaning of a word can be described by the set of textual contexts in which it appears (*Distributional Hypothesis* (Harris, 1964)), and that words with similar vectors are semantically related. In our setting, the goal is to find a semantic space model able to capture the notion of *frame* – i.e. the property of *"being characteristic of a frame"*. In such a model, an unknown LU is induced by first computing the similarity between its vector and the vectors of the existing frames, and then assigning the LU to the frame with the highest similarity.

3.1 Assigning unknown LUs to frames

In our model, a LU l is represented by a vector $\vec{l}$ whose dimensions represent the set of contexts C of the semantic space. The value of each dimension is given by the co-occurrence value of the LU with a contextual feature $c \in C$, computed over a large corpus using an association measure. We experiment with two different association measures: normalized frequency and pointwise mutual information. We approximate these measures by using Maximum Likelihood Estimation, as follows:

$$F(l,c) =_{MLE} \frac{|l,c|}{|*,*|}$$
$$MI(l,c) =_{MLE} \frac{|l,c||*,*|}{|*,c||l,*|} \tag{1}$$

where $|l,c|$ denotes the co-occurrence counts of the pair (l,c) in the corpus, $|*,c| = \sum_{l \in L} |l,c|$, $|l,*| = \sum_{c \in C} |l,c|$ and finally $|*,*| = \sum_{l \in L, c \in C} |l,c|$.

A frame f is modeled by a vector $\vec{f}$, representing the distributional profile of the frame in the semantic space. We here assume that a frame can be fully described by the set of its lexical units F. We implement this intuition by computing $\vec{f}$ as the weighted centroid of the set F, as follows:

$$\vec{f} = \sum_{l \in F} w_{lf} * \vec{l} \tag{2}$$

where w_{lf} is a weighting factor, accounting for the relevance of a given lexical unit with respect to the frame, estimated as:

$$w_{lf} = \frac{|l|}{\sum_{l \in F} |l|} \tag{3}$$

where $|l|$ denotes the counts of l in the corpus. From a more cognitive perspective, the vector $\vec{f}$ represents the prototypical lexical unit of the frame.

Given the set of all frames $\mathcal{N}$ and an unknown lexical unit ul, we assign ul to the frame $fmax_{ul}$ which is distributionally most similar – i.e. we intuitively map an unknown lexical unit to the frame whose prototypical lexical unit $\vec{f}$ has the highest similarity with $\vec{ul}$:

$$fmax_{ul} = argmax_{f \in \mathcal{N}} sim_D(\vec{ul}, \vec{f}) \tag{4}$$

In our model, we used the traditional cosine similarity:

$$sim_{cos}(ul, f) = \frac{\vec{ul} \cdot \vec{f}}{|\vec{ul}| * |\vec{f}|} \tag{5}$$

3.2 Choosing the space

Different types of contexts C define spaces with different semantic properties. We are here looking for a space able to capture the properties which characterise a frame. The most relevant of these properties is that LUs in the same frame tend to be either co-occurring or substitutional words (e.g. *assassin/kill* or *assassinate/kill*) – i.e. they are either in paradigmatic and syntagmatic relation. In an ideal space, a high similarity value sim_D would be then given both to *assassinate/kill* and to *assassin/kill*. We explore three spaces which seem to capture the above property well:

Word-based space: Contexts are words appearing in a n-window of the lexical unit. Such spaces model a generic notion of *semantic relatedness*. Two LUs close in the space are likely to be related by some type of generic semantic relation, either paradigmatic (e.g. synonymy, hyperonymy, antonymy) or syntagmatic (e.g. meronymy, conceptual and phrasal association).[1]

Syntax-based space: Contexts are syntactic relations (e.g. *X-VSubj-man* where X is the LU), as described in (Padó, 2007). These spaces are good at modeling *semantic similarity*. Two LUs close in the space are likely to be in a paradigmatic relation, i.e. to be close in a is-a hierarchy (Budanitsky and Hirst, 2006; Lin, 1998; Padó, 2007). Indeed, as contexts are syntactic relations, targets with the same part of speech are much closer than targets of different types.

Mixed space: In a combination of the two above spaces, contexts are words connected to the LU by a dependency path of at most length n. Unlike word-based spaces, contexts are selected in a more principled way: only syntactically related words are contexts, while other (possibly noisy) material is filtered out. Unlike syntax-based spaces, the context c does not explicitly state the type of syntactic relation with the LU: this usually allows to capture both paradigmatic and syntagmatic relations.

4 WordNet-based model

In a departure from previous work, our WordNet-based model does not rely on standard WordNet similarity measures (Pedersen et al., 2004), as these measures can only be applied to *pairs* of words, while we here need to capture the meaning of whole frames, which typically consist of larger sets of LUs. Our intuition is that senses able to evoke a frame can be detected via WordNet, by jointly considering the WordNet synsets activated by *all* LUs of the frame.

We implement this intuition in a weakly-supervised model, where each frame f is represented as a set of specific sub-graphs of the WordNet

[1]See (Padó, 2007; Sahlgren, 2006) for an in depth analysis.

hyponymy hierarchy. As different parts of speech have different WordNet hierarchies, we build a sub-graph for each of them: S_f^n for nouns, S_f^v for verbs and S_f^a for adjectives.[2] These sub-graphs represent the lexical semantic properties characterizing the frame. An unknown LU ul of a given part of speech is assigned to the frame whose corresponding sub-graph is semantically most similar to one of the senses of ul:

$$fmax_{ul} = argmax_{f \in \mathcal{N}} sim_{WN}(ul, f) \quad (6)$$

where sim_{WN} is a WordNet-based similarity measure. In the following subsections we will describe how we build sub-graphs and model the similarity measure for the different part of speech.

Figure 1 reports an excerpt of the noun sub-graph for the frame PEOPLE_BY_AGE, covering the suitable senses of its nominal LUs $\{adult, baby, boy, kid, youngster, youth\}$. The relevant senses (e.g. sense 1 of *youth* out of the 6 potential ones) are generally selected, as they share the most specific generalizations in WordNet with the other words.

Nouns. To compute similarity for *nouns* we adopt *conceptual density* (*cd*) (Agirre and Rigau, 1996), a semantic similarity model previously applied to word sense disambiguation tasks.

Given a frame f and its set of nominal lexical units F_n, the nominal subgraph S_f^n is built as follows. All senses of all words in F_n are activated in WordNet. All hypernyms H_f^n of these senses are then retrieved. Every synset $\sigma \in H_f^n$ is given a *cd* score, representing the *density* of the WordNet sub-hierarchy rooted at σ in representing the set of nouns F_n. The intuition behind this model is that the larger the number of LUs in F_n that are generalized by σ is, the better it captures the lexical semantics intended by the frame f. Broader generalizations are penalized as they give rise to bigger hierarchies, not well correlated with the full set of targets F_n.

To build the final sub-graph S_f^n, we apply the greedy algorithm proposed by Basili and colleagues (2004). It first computes the set of WordNet synsets that generalize at least two LUs in F_n, and then selects the subset of most dense ones $S_f^n \subset H_f^n$ that cover F_n. If a LU has no common hypernym with other members of F_n, it is not represented in S_f^n, and its similarity is set to 0 . S_f^n disambiguates words in F_n as only the lexical senses with at least one hypernym in S_f^n are considered.

Figure 1 shows the nominal sub-graph automatically derived using conceptual density for the frame PEOPLE_BY_AGE. The word *boy* is successfully disambiguated, as its only hypernym in the sub-graph refers to its third sense (*a male human offspring*) which correctly maps to the given frame. Notice that this model departs from the first sense heuristics largely successful in word sense disambiguation: most frames in fact are characterized by non predominant senses. The only questionable disambiguation is for the word $adult$: the wrong sense (*adult mammal*) is selected. However, even in these cases, the *cd* values are very low (about 10^{-4}), so that they do not impact much on the quality of the resulting inference.

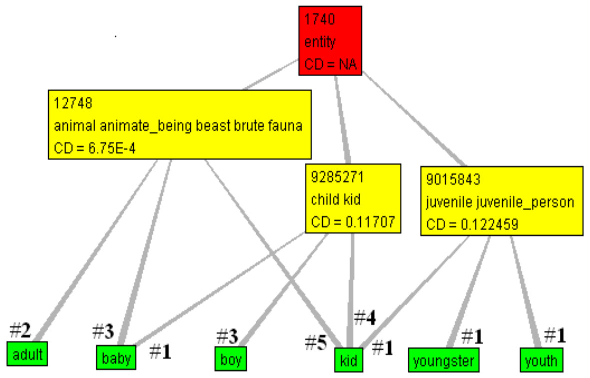


Figure 1: The noun sub-graph for the frame PEOPLE_BY_AGE as evoked by a subset of the words. Sense numbers #n refers to WordNet 2.0.

Using this model, LU induction is performed as follows. Given an unknown lexical unit ul, for each frame $f \in \mathcal{N}$ we first build the sub-graph S_f^n from the set $F_n \cup \{ul\}$. We then compute $sim_{WN}(f, ul)$ as the maximal *cd* of any synset $\sigma \in S_f^n$ that generalizes one of the lexical senses of ul. In the example *baby* would receive a score of 0.117 according to its first sense in WordNet 2.0 ("*baby,babe,infant*"). In a final step, we assign the LU to the most similar frame, according to Eq. 6

Verbs and Adjectives. As the conceptual density algorithm can be used only for nouns, we apply different similarity measures for verbs and adjectives.

[2]Our WordNet model does not cover the limited number of LUs which are not nouns, verbs or adjectives.

For *verbs* we exploit the co-hyponymy relation: the sub-graph S^v_f is given by all hyponyms of all verbs F_v in the frame f. Similarity $sim_{WN}(f, ul)$ is computed as follows:

$$sim_{WN}(ul, f) = \begin{cases} 1 & \textbf{iff } \exists K \subset F \text{ such that} \\ & |K| > \tau \textbf{ AND} \\ & \forall l \in K, l \text{ is a co-hyponym of } ul \\ \epsilon & \text{otherwise} \end{cases} \tag{7}$$

As for *adjectives*, WordNet does not provide a hyponymy hierarchy. We then compute similarity simply on the basis of the synonymy relation, as follows:

$$sim_{WN}(ul, f) = \begin{cases} 1 & \textbf{iff } \exists l \in F \text{ such that} \\ & l \text{ is a synonym of } ul \\ \epsilon & \text{otherwise} \end{cases} \tag{8}$$

5 Combined model

The methods presented so far use two independent information sources to induce LUs: distributional similarity sim_D and WordNet similarity sim_{WN}. We also build a joint model, leveraging both approaches: we expect the combination of different information to raise the overall performance. We here choose to combine the two approaches using a simple back-off model, that uses the WordNet-based model as a default and backs-off to the distributional one when no frame is proposed by the former. The intuition is that WordNet should guarantee the highest precision in the assignment, while distributional similarity should recover cases of low coverage.

6 Experiments

In this section we present a comparative evaluation of our models on the task of inducing LUs, in a leave-one-out setting over a reference gold standard.

6.1 Experimental Setup

Our gold standard is the FrameNet 1.3 database, containing 795 frames and a set L of 7,522 unique LUs (in all there are 10,196 LUs possibly assigned to more than one frame). Given a lexical unit $l \in L$, we simulate the induction task by executing a leave-one-out procedure, similarly to Burchardt and colleagues (2005). First, we remove l from all its original frames. Then, we ask our models to reassign it to the most similar frame(s) f, according to the similarity measure[3]. We repeat this procedure for all lexical units. Though our experiment is not completely realistic (we test over LUs already in FrameNet), it has the advantage of a reliable gold standard produced by expert annotators. A second, more realistic, small-scale experiment is described in Section 6.2.

We compute *accuracy* as the fraction of LUs in L that are correctly re-assigned to the original frame. Accuracy is computed at different levels k: a LU l is correctly assigned if its gold standard frame appears among the best-*k* frames f ranked by the model using the $sim(l, f)$ measure. As LUs can have more than one correct frame, we deem as correct an assignment for which at least one of the correct frames is among the best-*k*.

We also measure *coverage*, intended as the percentage of LUs that have been assigned to at least one frame by the model. Notice that when no sense preference can be found above the threshold ϵ, the WordNet-based model cannot predict any frame, thus decreasing coverage.

We present results for the following models and parametrizations (further parametrizations have revealed comparable performance).

Dist-word : the word-based space described in Section 3. Contextual features correspond to the set of the 4,000 most frequent words in the BNC.[4] The association measure between LUs and contexts is the pointwise mutual information. Valid contexts for LUs are fixed to a 20-window.

Dist-syntax : the syntax-based space described in Section 3. Context features are the 10,000 most frequent syntactic relations in the BNC[5]. As association measure we apply log-likelihood ratio (Dunning, 1993) to normalized frequency. Syntactic relations are extracted using the Minipar parser.

Dist-mixed : the mixed space described in Sec-

[3]In the distributional model, we recompute the centroids for each frame f in which the LU appeared, applying Eq. 2 to the set $F - \{l\}$.

[4]We didn't use the FrameNet corpus directly, as it is too small to obtain reliable statistics.

[5]Specifically, we use the minimum context selection function and the plain path value function described in Pado (2007).

tion 3. As for the *Dist-word* model, contextual features are 4,000 and pointwise mutual information is the association measure. The maximal dependency path length for selecting each context word is 3. Syntactic relations are extracted using Minipar.

WNet-full : the WordNet based model described in Section 4.

WNet-bsense : this model is computed as *WNet-full* but using only the most frequent sense for each LU as defined in WordNet.

Combined : the combined method presented in Section 5. Specifically, it uses *WNet-full* as a default and *Dist-word* as back-off.

Baseline-rnd : a baseline model, randomly assigning LUs to frames.

Baseline-mostfreq : a model predicting as best-*k* frames the most likely ones in FrameNet – i.e. those containing the highest number of LUs.

6.2 Experimental Results

Table 1 reports accuracy and coverage results for the different models, considering only 6792 LUs with frequency higher than 5 in the BNC, and frames with more than 2 lexical units (to allow better generalizations in all models). Results show that all our models largely outperform both baselines, achieving a good level of accuracy and high coverage. In particular, accuracy for the best-10 frames is high enoungh to support tasks such as the semi-automatic creation of new FrameNets. This claim is supported by a further task-driven experiment, in which we asked 3 annotators to assign 60 unknown LUs (from the Detour system log) to frames, with and without the support of the *Dist-word* model's predictions as suggestions[6]. We verified that our model guarantee an annotation speed-up of 25% – i.e. in average an annotator saves 25% of annotation time by using the system's suggestions.

Distributional vs. WordNet-based models. WordNet-based models are significantly better than distributional ones, for several reasons. First, distributional models acquire information only from the contexts in the corpus. As we do not use a FrameNet annotated corpus, there is no guarantee that the usage of a LU in the texts reflects exactly the semantic properties of the LU in FrameNet. In the extreme cases of polysemous LUs, it may happen that the textual contexts refer to senses which are not accounted for in FrameNet. In our study, we explicitly ignore the issue of polisemy, which is a notoriously hard task to solve in semantics spaces (see (Schütze, 1998)), as the occurrences of different word senses need to be clustered separately. We will approach the problem in future work. The WordNet-based model suffers from the problem of polisemy to a much lesser extent, as all senses are explicitly represented and separated in WordNet, including those related to the FrameNet gold standard.

A second issue regards data sparseness. The vectorial representation of LUs with few occurrences in the corpus is likely to be semantically incomplete, as not enough statistical evidence is available. Particularly skewed distributions can be found when some frames are very rarely represented in the corpus. A more in-depth descussion on these two issues is given later in this section.

Regarding the WordNet-based models, *WNet-full* in most cases outperforms *WNet-bsense*. The first sense heuristic does not seem to be as effective as in other tasks, such as Word Sense Disambiguation. Although sense preferences (or predominance) across two general purpose resources, such as WordNet and FrameNet, should be a useful hint, the conceptual density algorithm seems to produce better distributions (i.e. higher accuracy), especially when several solutions are considered. Indeed, for many LUs the first WordNet sense is not the one represented in the FrameNet database.

As for distributional models, results show that the *Dist-word* model performs best. In general, syntactic relations (*Dist-syntax* model) do not help to capture frame semantic properties better than a simple window-based approach. This seems to indicate that LUs in a same frame are related both by paradigmatic and syntagmatic relations, in accordance to the definition given in Section 3.2 – i.e. they are mostly semantically *related*, but not *similar*.

Coverage. Distributional models show a coverage 15% higher than WordNet-based ones. Indeed, as far as corpus evidence is available (i.e. the unknown LU appears in the corpus), distributional methods are always able to predict a frame. WordNet-based mod-

[6]For this purpose, the dataset is evenly split in two parts.

MODEL	B-1	B-2	B-3	B-4	B-5	B-6	B-7	B-8	B-9	B-10	COVERAGE
Dist-word	0.27	0.36	0.42	0.46	0.49	0.51	0.53	0.55	0.56	0.57	95%
Dist-syntax	0.22	0.29	0.34	0.38	0.41	0.44	0.46	0.48	0.50	0.51	95%
Dist-mixed	0.25	0.35	0.40	0.44	0.47	0.49	0.51	0.53	0.54	0.56	95%
WNet-full	0.47	0.59	0.65	0.69	0.72	0.73	0.75	0.76	0.77	0.78	80%
WNet-bsense	0.52	0.61	0.64	0.66	0.67	0.68	0.69	0.69	0.70	0.70	72%
Combined	0.43	0.54	0.60	0.64	0.66	0.68	0.70	0.71	0.72	0.73	95%
Baseline-rnd	*0.02*	*0.03*	*0.05*	*0.06*	*0.08*	*0.10*	*0.11*	*0.12*	*0.14*	*0.15*	
Baseline-mostfreq	*0.02*	*0.05*	*0.07*	*0.08*	*0.10*	*0.11*	*0.13*	*0.14*	*0.15*	*0.17*	

Table 1: Accuracy and coverage of different models on best-*k* ranking with frequency threshold 5 and frame threshold 2

els cannot make predictions in two specific cases. First, when the LU is not present in WordNet. Second, when the function sim_{WN} does not has sufficient relational information to find a similar frame. This second factor is particularly evident for adjectives, as Eq. 8 assigns a frame only when a synonym of the unknown LU is found. It is then not surprising that 68% of the missed assignment are indeed adjectives.

Results for the *Combined* model suggest that the integration of distributional and WordNet-based methods can offer a viable solution to the coverage problem, as it achieves an accuracy comparable to the pure WordNet approaches, while keeping the coverage high.

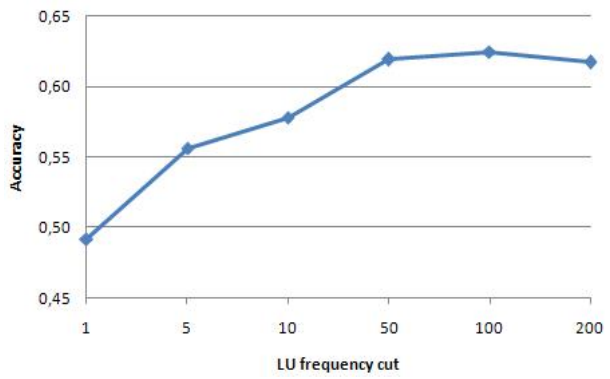


Figure 2: *Dist-word* model accuracy at different LU frequency cuts.

Data Sparseness. A major issue when using distributional approaches is that words with low frequency tend to have a very sparse non-meaningful representation in the vector space. This highly impacts on the accuracy of the models. To measure the impact of data sparseness, we computed the accuracy at different frequency cuts – i.e. we exclude LUs below a given frequency threshold from centroid computation and evaluation. Figure 2 reports the results for best-*10* assignment at different cuts, for the *Dist-word* model. As expected, accuracy improves by excluding infrequent LUs. Only at a frequency cut of 200 performance becomes stable, as statistical evidence is enough for a reliable prediction. Yet, in a real setting the improvement in accuracy implies a lower coverage, as the system would not classify LUs below the threshold. For example, by discarding LUs occurring less than 200 times in the corpus, we obtain a +0.12 improvement in accuracy, but the coverage decreases to 57%. However, uncovered LUs are also the most rare ones and their relevance in an application may be negligible.

Lexical Semantics, Ambiguity and Plausible Assignments. The overall accuracies achieved by our methods are "pessimistic", in the sense that they should be intended as lower-bounds. Indeed, a qualitative analysis of erroneous predictions reveals that in many cases the frame assignments produced by the models are semantically plausible, even if they are considered incorrect in the leave-one-out test. Consider for example the LU *guerrilla*, assigned in FrameNet to the frame PEOPLE BY VOCATION. Our mixed model proposes as two most similar frames MILITARY and TERRORISM, which could still be considered plausible assignment. The same holds for the LU *caravan*, for which the most similar frame is VEHICLE, while in FrameNet the LU is assigned only to the frame BUILDINGS. These cases are due to the low FrameNet coverage, i.e LUs are not fully annotated and they appear only in a subset of their potential frames. The real accuracy of our models is therefore expected to be higher.

To explore the issue, we carried out a qualitative analysis of 5 words (i.e. *abandon.v*, *accuse.v*, *body.n*, *charge.v* and *partner.n*). For each of them, we randomly picked 60 sentences from the BNC corpus, and asked two human annotators to assign to the correct frame the occurrence of the word in the given sentence. For 2 out of 5 words, no frame could be found for most of the sentences, suggesting that the most frequent frames for these words were missing from FrameNet[7]. We can then conclude that 100% accuracy cannot be considered as the upper-bound of our experiment, as word usage in texts is not well reflected in the FrameNet modelling.

Further experiments. We also tested our models on a realistic gold-standard set of 24 unknown LUs extracted from the SemEval-2007 corpus (Baker et al., 2007). These are words not present in FrameNet 1.3 which have been assigned by human annotators to an existing frame[8]. *WNet-full* achieves an accuracy of 0.25 for best-1 and 0.69 for best-10, with a coverage of 67%. A qualitative analysis showed that the lower performance wrt to our main experiment is due to higher ambiguity of the LUs (e.g. we assign *tea* to SOCIAL_EVENT instead of FOOD).

Comparison to other approaches. We compare our models to the system presented by Johansson and Nugues (2007) and Burchardt and colleagues (2005). Johansson and Nugues (2007) evaluate their machine learning system using 7,000 unique LUs to train the Support Vector Machine, and the remaining LUs as test. They measure accuracy at different coverage levels. At 80% coverage accuracy is about 0.42, 10 points below our best WordNet-based system. At 90% coverage, the system shows an accuracy below 0.10 and is significantly outperformed by both our distributional and combined methods. These results confirm that WordNet-based approaches, while being highly accurate wrt distributional ones, present strong weaknesses as far as coverage is concerned. Furthermore, Johansson and Nugues (2007) show that their machine learning approach outperforms a simple approach based on WordNet similarity: thus, our results indirectly prove that our WordNet-based method is more effective than the application of the similarity measure presented in (Pedersen et al., 2004).

We also compare our results to those reported by Burchardt and colleagues (2005) for Detour. Though the experimental setting is slightly different (LU assignment is done at the text-level), they use the same gold standard and leave-one-out technique, reporting a best-1 accuracy of 0.38 and a coverage of 87%. Our WordNet-based models significantly outperform Detour on best-1 accuracy, at the cost of lower coverage. Yet,our *combined* model is significantly better both on accuracy (+5%) and coverage (+8%). Also, in most cases Detour cannot predict more than one frame (best-1), while our accuracies can be improved by relaxing to any best-k level.

7 Conclusions

In this paper we presented an original approach for FrameNet LU induction. Results show that models combining distributional and WordNet information offer the most viable solution to model the notion of frame, as they allow to achieve a reasonable trade-off between accuracy and coverage. We also showed that in contrast to previous work, simple semantic spaces are more helpful than complex syntactic ones. Results are accurate enough to support the creation and the development of new FrameNets.

As future work, we will evaluate new types of spaces (e.g. dimensionality reduction methods) to improve the generalization capabilities of the space models. We will also address the data sparseness issue, by testing smoothing techniques to better model low frequency LUs. Finally, we will implement the presented models in a complex architecture for semi-supervised FrameNets development, both for specializing the existing English FrameNet in specific domains, and for creating new FrameNets in other languages.

Acknowledgements

This work has partly been funded by the German Research Foundation DFG (grant PI 154/9-3). Thanks to Richard Johansson and Aljoscha Burchardt for providing the data of their systems.

[7]Note that the need of new frames to account for semantic phenomena in free texts has been also demonstrated by the SemEval-2007 competition.

[8]The set does not contain 4 LUs which have no frame in FrameNet.

References

E. Agirre and G. Rigau. 1996. Word Sense Disambiguation using Conceptual Density. In *Proceedings of COLING-96*, Copenhagen, Denmark.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of COLING-ACL*, Montreal, Canada.

Collin Baker, Michael Ellsworth, and Katrin Erk. 2007. SemEval-2007 Task 19: Frame Semantic Structure Extraction. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 99–104, Prague, Czech Republic, June.

Roy Bar-Haim, Idan Szpektor, and Oren Glickman. 2005. Definition and Analysis of Intermediate Entailment Levels. In *ACL-05 Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, Ann Arbor, Michigan.

R. Basili, M. Cammisa, and F.M. Zanzotto. 2004. A semantic similarity measure for unsupervised semantic disambiguation. In *Proceedings of LREC-04*, Lisbon, Portugal.

Alexander Budanitsky and Graeme Hirst. 2006. Evaluating WordNet-based measures of semantic distance. *Computational Linguistics*, 32(1):13–47.

Aljoscha Burchardt and Anette Frank. 2006. Approximating Textual Entailment with LFG and FrameNet Frames. In *Proceedings of PASCAL RTE2 Workshop*.

Aljoscha Burchardt, Katrin Erk, and Anette Frank. 2005. A WordNet Detour to FrameNet. In *Sprachtechnologie, mobile Kommunikation und linguistische Resourcen*, volume 8 of *Computer Studies in Language and Speech*. Peter Lang, Frankfurt/Main.

Aljoscha Burchardt, Katrin Erk, Anette Frank, Andrea Kowalski, Sebastian Padó, and Manfred Pinkal. 2006. The SALSA corpus: a German corpus resource for lexical semantics. In *Proceedings of LREC*, Genova, Italy.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 18(1):61–74.

Charles J. Fillmore. 1985. Frames and the semantics of understanding. *Quaderni di Semantica*, 4(2):222–254.

K. Garoufi. 2007. Towards a better understanding of applied textual entailment: Annotation and evaluation of the rte-2 dataset. *M.Sc. thesis*, saarland university.

Zellig Harris. 1964. Distributional structure. In Jerrold J. Katz and Jerry A. Fodor, editors, *The Philosophy of Linguistics*, New York. Oxford University Press.

Richard Johansson and Pierre Nugues. 2007. Using WordNet to extend FrameNet coverage. In *Proceedings of the Workshop on Building Frame-semantic Resources for Scandinavian and Baltic Languages, at NODALIDA*, Tartu, Estonia, May 24.

Dekang Lin. 1998. Automatic retrieval and clustering of similar word. In *Proceedings of COLING-ACL*, Montreal, Canada.

Sebastian Padó. 2007. *Cross-Lingual Annotation Projection Models for Role-Semantic Information*. Saarland University.

M. Palmer, D. Gildea, and P. Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1).

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. WordNet::Similarity - Measuring the Relatedness of Concept. In *Proc. of 5th NAACL*, Boston, MA.

Magnus Sahlgren. 2006. *The Word-Space Model*. Department of Linguistics, Stockholm University.

G. Salton, A. Wong, and C. Yang. 1975. A vector space model for automatic indexing. *Communications of the ACM*, 18:613620.

Hinrich Schütze. 1998. Automatic Word Sense Discrimination. *Computational Linguistics*, 24(1):97–124.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of EMNLP-CoNLL*, pages 12–21, Prague.

C. Subirats and M. Petruck. 2003. Surprise! Spanish FrameNet! In *Proceedings of the Workshop on Frame Semantics at the XVII. International Congress of Linguists*, Prague.

Multimodal Subjectivity Analysis of Multiparty Conversation

Stephan Raaijmakers
TNO Information and
Communication Technology
Delft, The Netherlands
stephan.raaijmakers@tno.nl

Khiet Truong
TNO Defense, Security and Safety
Soesterberg, The Netherlands
khiet.truong@tno.nl

Theresa Wilson
School of Informatics
University of Edinburgh
Edingburgh, UK
twilson@inf.ed.ac.uk

Abstract

We investigate the combination of several sources of information for the purpose of subjectivity recognition and polarity classification in meetings. We focus on features from two modalities, transcribed words and acoustics, and we compare the performance of three different textual representations: words, characters, and phonemes. Our experiments show that character-level features outperform word-level features for these tasks, and that a careful fusion of all features yields the best performance. [1]

1 Introduction

Opinions, sentiments and other types of subjective content are an important part of any meeting. Meeting participants express pros and cons about ideas, they support or oppose decisions, and they make suggestions that may or may not be adopted. When recorded and archived, meetings become a part of the organizational knowledge, but their value is limited by the ability of tools to search and summarize meeting content, including subjective content. While progress has been made on recognizing primarily objective meeting content, for example, information about the topics that are discussed (Hsueh and Moore, 2006) and who is assigned to work on given tasks (Purver et al., 2006), there has been fairly little work specifically directed toward recognizing subjective content.

In contrast, there has been a wealth of research over the past several years on automatic subjectivity and sentiment analysis in text, including on-line media. Partly inspired by the rapid growth of social media, such as blogs, as well as on-line news and reviews, researchers are now actively addressing a wide variety of new tasks, ranging from blog mining (e.g., finding opinion leaders in an on-line community), to reputation management (e.g. finding negative opinions about a company on the web), to opinion-oriented summarization and question answering. Yet many challenges remain, including how best to represent and combine linguistic information for subjectivity analysis. With the additional modalities that are present when working with face-to-face spoken communication, these challenges are even more pronounced.

The work in this paper focuses on two tasks: (1) recognizing subjective utterances and (2) discriminating between positive and negative subjective utterances. An utterance may be subjective because the speaker is expressing an opinion, because the speaker is discussing someone else's opinion, or because the speaker is eliciting the opinion of someone else with a question.

We approach the above tasks as supervised machine learning problems, with the specific goal of finding answers to the following research questions:

- Given a variety of information sources, such as text arising from (transcribed) speech, phoneme representations of the words in an utterance, and acoustic features extracted from

[1] This work was supported by the Dutch BSIK-project MultimediaN, and the European IST Programme Project FP6-0033812. This paper only reflects the authors' views and funding agencies are not liable for any use that may be made of the information contained herein.

the audio layer, which of these sources are particularly valuable for subjectivity analysis in multiparty conversation?

- Does the combination of these sources lead to further improvement?
- What are the optimal representations of these information sources in terms of feature design for a machine learning component?

A central tenet of our approach is that subword representations, such as *character* and *phoneme n-grams*, are beneficial for the tasks at hand.

2 Subword Features

Previous work has demonstrated that textual units below the word level, such as character n-grams, are valuable sources of information for various text classification tasks. An example of character n-grams is the set of 3-grams {`#se, sen, ent, nti, tim, ime, men, ent, nt#, t#a, #an, ana, nal, aly, lys, ysi, sis, is#`} for the two-word phrase *sentiment analysis*. The special symbol # represents a word boundary. While it is not directly obvious that there is much information in these truncated substrings, character n-grams have successfully been used for fine-grained classification tasks, such as named-entity recognition (Klein et al., 2003) and subjective sentence recognition (Raaijmakers and Kraaij, 2008), as well as a variety of document-level tasks (Stamatatos, 2006; Zhang and Lee, 2006; Kanaris and Stamatatos, 2007).

The informativeness of these low-level features comes in part from a form of *attenuation* (Eisner, 1996): a slight abstraction of the underlying data that leads to the formation of string equivalence classes. For instance, words in a sentence will invariably share many character n-grams. Since every unique character n-gram in an utterance constitutes a separate feature, this leads to the formation of string classes, which is a form of abstraction. For example, Zhang and Lee (2006) investigate similar subword representations, called key substring group features. By compressing substrings in a corpus in a trie (a prefix tree), and labeling entire sets of distributionally equivalent substrings with one group label, an attenuation effect is obtained that proves very beneficial for a number of text classification tasks.

Aside from attenuation effects, character n-grams, especially those that represent word boundaries, have additional benefits. Treating word boundaries as characters captures micro-phrasal information: short strings that express the transition of one word to another. Stemming occurs naturally within the set of initial character n-grams of a word, where the suffix is left out. Also, some part-of-speech information is captured. For example, the modals *could, would, should* can be represented by the 4-gram, `ould`, and the set of adverbs ending in *-ly* can be represented by the 3-gram `ly#`.

A challenging thought is to extend the use of n-grams to the level of phonemes, which comprise the first symbolic level in the process of sound to grapheme conversion. If n-grams of phonemes compare favorably to word n-grams for the purpose of sentiment classification, then significant speedups can be obtained for online sentiment classification, since tokenization of the raw speech signal can make a halt at the phoneme level.

3 Data

For this work we use 13 meetings from the AMI Meeting Corpus (Carletta et al., 2005). Each meeting has four participants and is approximately 30 minutes long. The participants play specific roles (e.g., Project Manager, Marketing Expert) and together function as a design team. Within the set of 13 meetings, there are a total of 20 participants, with each participant taking part in two or three meetings as part of the same design team. Meetings with the same set of participants represent different stages in the design process (e.g., Conceptual Design, Detailed Design).

The meetings used in the experiments have been annotated for subjective content using the AMIDA annotation scheme (Wilson, 2008). Table 1 lists the types of annotations that are marked in the data. There are three main categories of annotations, *subjective utterances*, *subjective questions*, and *objective polar utterances*. A subjective utterance is a span of words (or possibly sounds) where a *private state* is being expressed either through choice of words or prosody. A private state (Quirk et al., 1985) is an internal mental or emotional state, including opinions, beliefs, sentiments, emotions, evaluations, uncertainties, and speculations, among others. Although typically when a private state is expressed it is the private state of the speaker, as in example (1) below, an utterance may also be subjective because the speaker is talking about the private state of someone else. For example, in (2) the negative opinion attributed to the company is what makes the utterance subjective.

> (1) Finding them is really a pain, you know
> (2) The company's decided that teletext is outdated

Subjective questions are questions in which the speaker is eliciting the private state of someone else. In other words, the speaker is asking about what someone else thinks, feels, wants, likes, etc., and the speaker is expecting a response in which the other person expresses what he or she thinks, feels, wants, or likes. For example, both (3) and (4) below are subjective questions.

> (3) Do you like the large buttons?
> (4) What do you think about the large buttons?

Objective polar utterances are statements or phrases that describe positive or negative factual information about something without conveying a private state. The sentence *The camera broke the first time I used it* gives an example of negative factual information; generally, something breaking the first time it is used is not good.

For the work in this paper, we focus on recognizing subjectivity in general and distinguishing between positive and negative subjective utterances. *Positive subjective* utterances are those in which any of the following types of private states are expressed: agreements, positive sentiments, positive suggestions, arguing for something, beliefs from which positive sentiments can be inferred, and positive responses to subjective questions. *Negative subjective* utterances express private states that are the opposite of those represented by the positive subjective category: disagreements, negative sentiments, negative suggestions, arguing against something, beliefs from which negative sentiments can be inferred, and negative responses to subjective questions. Example (5) below contains two positive subjective utterances and one negative subjective utterance. Each annotation is indicated by a pair of angle brackets.

Table 1: AMIDA Subjectivity Annotation Types

Subjective Utterances
positive subjective
negative subjective
positive and negative subjective
uncertainty
other subjective
subjective fragment
Subjective Questions
positive subjective question
negative subjective question
general subjective question
Objective Polar Utterances
positive objective
negative objective

> (5) Um ⟨**POS-SUBJ** it's very easy to use⟩. Um ⟨**NEG-SUBJ** but unfortunately it does lack the advanced functions⟩ ⟨**POS-SUBJ** which I I quite like having on the controls⟩.

The *positive and negative subjective* category is for marking cases of positive and negative subjectivity that are so closely interconnected that it is difficult or impossible to separate the two. For example, (6) below is marked as both positive and negative subjective.

> (6) Um ⟨**POS-AND-NEG-SUBJ** they've also suggested that we um we only use the remote control to control the television, not the VCR, DVD or anything else⟩.

In (Wilson, 2008), agreement is measured for each class separately at the level of dialogue act segments. If a dialogue act overlaps with an annotation of a particular type, then the segment is considered to be labelled with that type. Table 2 gives the Kappa (Cohen, 1960) and % agreement for subjective segments, positive and negative subjective segments,[2] and subjective questions.

[2]A positive subjective segment is any dialogue act segment that overlaps with a positive subjective utterance or a positive-and-negative subjective utterance. The negative subjective segments are defi ned similarly.

Table 2: Interannotator agreement for the AMIDA subjectivity annotations

	Kappa	% Agree
Subjective	0.56	79
Pos Subjective	0.58	84
Neg Subjective	0.62	92
Subjective Question	0.56	95

4 Experiments

We conduct two sets of classification experiments. For the first set of experiments (Task 1), we automatically distinguish between subjective and non-subjective utterances. For the second set of experiments (Task 2), we focus on distinguishing between positive and negative subjective utterances. For both tasks, we use the manual dialogue act segments available as part of the AMI Corpus as the unit of classification. For Task 1, a segment is considered subjective if it overlaps with either a subjective utterance or subjective question annotation. For Task 2, the segments being classified are those that overlap with positive or negative subjective utterances. For this task, we exclude segments that are both positive and negative. Although limiting the set of segments to be classified to just those that are positive or negative makes the task somewhat artificial, it also allows us to focus in on the performance of features specifically for this task.[3] We use 6226 subjective and 8707 non-subjective dialog acts for Task 1 (with an average duration of 1.9s, standard deviation of 2.0s), and 3157 positive subjective and 1052 negative subjective dialog acts for Task 2 (average duration of 2.6s, standard deviation of 2.3s).

The experiments are performed using 13-fold cross validation. Each meeting constitutes a separate fold for testing, e.g., all the segments from meeting 1 make up the test set for fold 1. Then, for a given fold, the segments from the remaining 12 meetings are used for training and parameter tuning, with roughly a 85%, 7%, and 8% split between training, tuning, and testing sets for each fold. The assignment to training versus tuning set was random, with the only constraint being that a segment could only be in the tuning set for one fold of the data.

[3]In practice, this excludes about 7% of the positive/negative segments.

The experiments we perform involve two steps. First, we train and optimize a classifier for each type of feature using BoosTexter (Schapire and Singer, 2000) AdaBoost.MH. Then, we investigate the performance of all possible combinations of features using linear combinations of the individual feature classifiers.

4.1 Features

The two modalities that are investigated, prosodic, and textual, are represented by four different sets of features: prosody (PROS), word n-grams (WORDS), character n-grams (CHARS), and phoneme n-grams (PHONES).

Based on previous research on prosody modelling in a meeting context (Wrede and Shriberg, 2003) and on the literature in emotion research (Banse and Scherer, 1996) we extract PROS features that are mainly based on pitch, energy and the distribution of energy in the long-term averaged spectrum (LTAS) (see Table 3). These features are extracted at the word level and aggregated to the dialogue-act level by taking the average over the words per dialogue act. We then normalize the features per speaker per meeting by converting the raw feature values to z-scores ($z = (x - \mu)/\sigma$).

Table 3: Prosodic features used in experiments.

pitch	mean, standard deviation, minimum, maximum, range, mean absolute slope
intensity (energy)	mean, standard deviation, minimum, maximum, range, RMS energy
distribution energy in LTAS	slope, Hammerberg index, centre of gravity, skewness

The textual features, WORDS and CHARS, and the PHONES features are based on a manual transcription of the speech. The PHONES were produced through dictionary lookup on the words in the reference transcription. Both CHARS and PHONES representations include word boundaries as informative tokens. The textual features for a given segment are simply all the WORDS/CHARS/PHONES in that segment. Selection of n-grams is performed by the learning algorithm.

4.2 Single Source Classifiers

We train four single source classifiers using BoosTexter, one for each type of feature. For the WORDS, CHARS, and PHONES, we optimize the classifier by performing a grid search over the parameter space, varying the number of rounds of boosting (100, 500, 1000, 2000, 5000), the length of the n-gram (1, 2, 3, 4, 5), and the type of n-gram. BoosTexter can be run with three different n-gram configurations: n-gram, s-gram, and f-gram. For the default configuration (n-gram), BoosTexter searches for n-grams up to length n. For example, if $n = 3$, BoosTexter will consider 1-grams, 2-grams, and 3-grams. For the s-gram configuration, BoosTexter will in addition consider sparse n-grams (i.e., n-grams containing wildcards), such as *the * idea*. For the f-gram configuration, BoosTexter will only consider n-grams of a maximum fixed length, e.g., if $n = 3$ BoosTexter will only consider 3-grams. For the PROS classifier, only the number of rounds of boosting was varied. The parameters are selected for each fold separately; the parameter set that produces the highest subjective F_1 score on the tuning set for Task 1, and the highest positive subjective F_1 score for Task 2, is used to train the final classifier for that fold.

4.3 Classifier combination

After the single source classifiers have been trained, they have to be combined into an aggregate classifier. To this end, we decided to apply a simple linear interpolation strategy. Linear interpolation of models is the weighted combination of simple models to form complex models, and has its roots in generative language models (Jelinek and Mercer, 1980). (Raaijmakers, 2007) has demonstrated its use for discriminative machine learning.

In the present binary class setting, BoosTexter produces two decision values, one for every class. For every individual single-source classifier (i.e., PROS, WORDS, CHARS and PHONES), separate weights are estimated that are applied to the decision values for the two classes produced by these classifiers. These weights express the relative importance of the single-source classifiers.

The prediction of an aggregate classifier for a class c is then simply the sum of all weights for all participating single-source classifiers applied to the decision values these classifiers produce for this class. The class with the maximum score wins, just as in the simple non-aggregate case.

Formally, then, this linear interpolation strategy finds for n single-source classifiers n interpolation weights $\lambda_1, \dots \lambda_n$ that minimize the empirical loss (measured by a loss function $\mathcal{L}$), with λ_j the weight of classifier j ($\lambda \in [0,1]$), and $C_c^j(x_i)$ the decision value of class c produced by classifier j for datum x_i (a feature vector). The two classes are denoted with $0, 1$. The true class for datum x_i is denoted with $\hat{x}_i$. The loss function is in our case based on subjective F-measure (Task 1) or positive subjective F-measure (Task 2) measured on heldout development training and test data.

The aggregate prediction $\tilde{x_i}$ for datum x_i on the basis of n single-source classifiers then becomes

$$\tilde{x_i} = \arg\max_c \left(\sum_{j=1}^{n} \lambda_j \cdot C_{c=0}^j(x_i), \sum_{j=1}^{n} \lambda_j \cdot C_{c=1}^j(x_i)\right) \tag{1}$$

and the lambdas are defined as

$$\lambda_j^n = \arg \min_{\lambda_j^n \subset [0,1]} \sum_i^k \mathcal{L}(\hat{x}_i, \tilde{x}_i; \lambda_j, \dots, \lambda_n) \tag{2}$$

The search process for these weights can easily be implemented with a simple grid search over admissible ranges.

In the experiments described below, we investigate all possible combinations of the four different sets of features (PROS, WORDS, CHARS, and PHONES) to determine which combination yields the best performance for subjectivity and subjective polarity recognition.

5 Results and Discussion

Results for the two tasks are given in Tables 4 and 5 and in Figures 1 and 2. We use two baselines, listed at the top of each table. The bullets in a given row indicate the features that are being evaluated for a given experiment. In Table 4, subjective F_1, recall, and precision are reported as well as overall accuracy. In Table 4, the F_1, recall, and precision scores are for the positive subjective class. All values in the tables are averages over the 13 folds.

Table 4: Results Task 1: Subjective vs. Non-Subjective.

	PROS	WORDS	CHARS	PHONES	F_1	PREC	REC	ACC
BASE-SUBJ	always chooses subjective class				60.3	43.4	100	43.4
BASE-RAND	randomly chooses a class based on priors				41.8	42.9	41.3	50.6
single	●				54.6	55.3	54.5	63.1
		●			60.5	68.5	54.5	71.0
			●		61.7	67.5	57.2	71.1
				●	60.3	66.4	55.5	70.2
double	●	●			63.9	72.1	57.6	73.4
	●		●		65.6	71.9	60.3	74.0
	●			●	64.6	72.3	58.4	73.7
		●	●		66.2	73.8	60.1	74.9
		●		●	65.2	73.2	58.8	74.3
			●	●	66.1	72.8	60.7	74.5
triple	●	●	●		66.5	74.3	60.3	75.1
	●	●		●	65.5	73.5	59.0	74.5
	●		●	●	66.5	73.3	60.8	74.8
		●	●	●	66.9	74.3	60.9	75.3
quartet	●	●	●	●	67.1	74.5	61.2	75.4

Table 5: Results Task 2: Positive Subjective vs. Negative Subjective.

	PROS	WORDS	CHARS	PHONES	F_1	PREC	REC	ACC
BASE-POS-SUBJ	always chooses positive subjective class				85.6	75.0	100	75.0
BASE-RAND	randomly chooses a class based on priors				75.1	74.4	76.1	62.4
single	●				84.8	74.8	98.1	73.9
		●			85.6	79.6	93.1	76.8
			●		85.9	81.9	90.5	78.0
				●	85.5	80.5	91.3	77.0
double	●	●			88.7	83.0	95.4	81.9
	●		●		88.7	83.1	95.1	81.8
	●			●	88.5	83.3	94.4	81.6
		●	●		89.5	84.2	95.7	83.3
		●		●	89.2	83.7	95.5	82.8
			●	●	89.0	84.2	94.6	82.6
triple	●	●	●		89.6	84.0	96.1	83.4
	●	●		●	89.3	83.6	95.8	82.8
	●		●	●	89.2	83.7	95.5	82.7
		●	●	●	89.8	84.4	96.0	83.8
quartet	●	●	●	●	89.9	84.4	96.2	83.8

It is quite obvious that the combination of different sources of information is beneficial, and in general, the more information the better the results. The best performing classifier for Task 1 uses all the features, achieving a subjective F_1 of 67.1. For Task 2, the best performing classifier also uses all the features, although it does not perform significantly better than the classifier using only WORDS, CHARS, and PHONES.[4] This classifier achieves a positive-subjective F_1 of 89.9.

We measured the effects of adding more information to the single source classifiers. These results are listed in Table 6. Of the various feature types, prosody seems to be the least informative for both subjectivity and polarity classification. In addition to producing the single-source classifier with the lowest performance for both tasks, Table 6 shows that when prosody is added, of all the features it is least likely to yield significant improvements.

Throughout the experiments, adding an additional type of textual feature always yields higher results. In all cases but two, these improvements are significant. The best performing of the features are the character n-grams. Of the single-source experiments, the character n-grams achieve the best performance, with significant improvements in F_1 over the other single-source classifiers for both Task 1 and Task 2. Also, adding character n-grams to other feature combinations always gives significant improvements in performance.

An obvious question that remains is what the effect is of classifier interpolation on the results. To answer this question, we conducted two additional experiments for both tasks. First, we investigated the performance of an uninterpolated combination of the four single-source classifiers. In essence, this combines the separate feature spaces without explicitly weighting them. Second, we investigated the results of training a single BoosTexter model using all the features, essentially merging all feature spaces

[4]We measured significance with the non-parametric Wilcoxon signed rank test, $p < 0.05$.

Table 6: Addition of features separately (for Task 1 and 2): '+' for a row-column pair (r, c) means that the addition of column feature c to the row features r significantly improved r's F_1; '-' indicates no significant improvement; 'X' means 'not applicable'

	+PROS		+ WORDS		+CHARS		+PHONES	
Task	1	2	1	2	1	2	1	2
PROS	X	X	+	+	+	+	+	+
WORDS	-	+	X	X	+	+	+	+
CHARS	-	+	-	+	X	X	-	+
PHONES	-	+	+	+	+	+	X	X
PROS+WORDS	X	X	X	X	+	+	+	+
PROS+CHARS	X	X	+	+	X	X	+	+
PROS+PHONES	X	X	+	+	+	+	X	X
WORDS+CHARS	+	-	X	X	X	X	+	+
WORDS+PHONES	+	-	X	X	+	+	X	X
CHARS+PHONES	+	+	+	+	X	X	X	X
PROS+WORDS+CHARS	X	X	X	X	X	X	+	+
PROS+WORDS+PHONES	X	X	X	X	+	+	X	X
PROS+CHARS+PHONES	X	X	+	+	X	X	X	X
WORDS+CHARS+PHONES	+	-	X	X	X	X	X	X

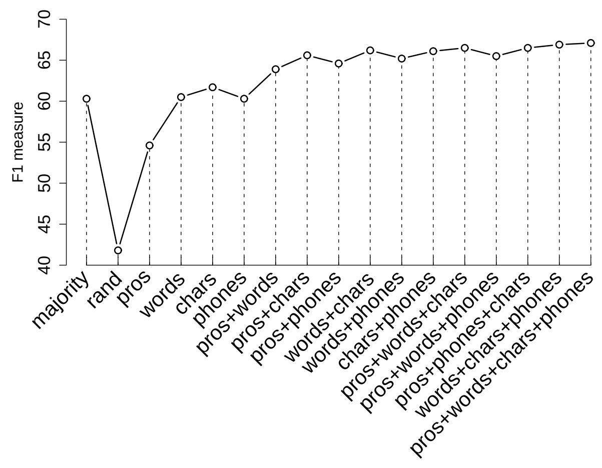


Figure 1: *Results (F_1) experiment 1: subjective vs. non-subjective.*

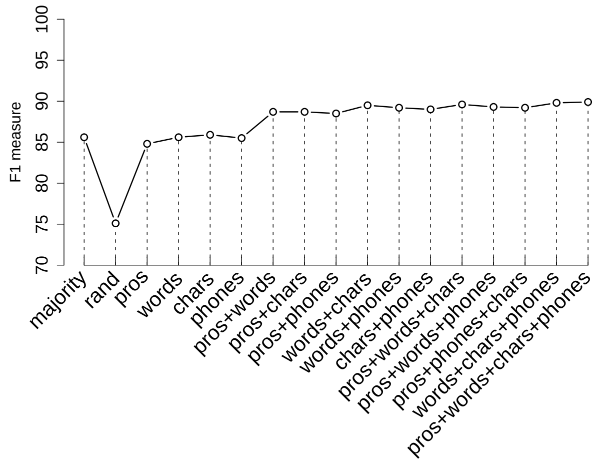


Figure 2: *Results (F_1) experiment 2: positive subjective vs. negative subjective.*

into one agglomerate feature space. The results for these experiments are given in Table 7, along with the results from the all-feature interpolated classification for comparison.

The results in Table 7 show that interpolation outperforms both the unweighted and single-model combinations for both tasks. For Task 1, the effect of interpolation compared to a single model is marginal (a .03 point difference in F_1). However, compared to the uninterpolated combination, interpolation gives a clear 3.1 points improvement of F_1. For Task 2, interpolation outperforms both the uninterpolated and single-model classifiers, with 2 and 3 points improvements in F_1, respectively.

6 Related Work

Previous work has demonstrated that textual units below the word level, such as character n-grams, are valuable sources of information. Character-level models have successfully been used for named-entity recognition (Klein et al., 2003), predicting authorship (Keselj et al., 2003; Stamatatos, 2006), text categorization (Zhang and Lee, 2006), web page genre identification (Kanaris and Stamatatos, 2007), and sentence-level subjectivity recognition (Raaijmakers and Kraaij, 2008) In spoken-language data, Hsueh (2008) achieves good results using chains of phonemes to automatically segment meetings according to topic. However, to the best of our knowledge there has been no investigation to date on the

Table 7: Results of interpolated classifiers compared to uninterpolated and single-model classifiers for all features.

Task	Combination	ACC	REC	PREC	F_1
1	interpolated	75.4	61.2	74.5	67.1
	uninterpolated	73.0	58.7	70.6	64.0
	single model	74.7	62.1	72.7	66.8
2	interpolated	83.8	96.2	84.4	89.9
	uninterpolated	79.8	98.0	79.7	87.9
	single model	79.5	91.0	83.3	86.9

combination of character-level, phoneme-level, and word-level models for any natural language classification tasks.

In text, there has been a significant amount of research on subjectivity and sentiment recognition, ranging from work at the phrase level to work on classifying sentences and documents. Sentence-level subjectivity classification (e.g., (Riloff and Wiebe, 2003; Yu and Hatzivassiloglou, 2003)) and sentiment classification (e.g., (Yu and Hatzivassiloglou, 2003; Kim and Hovy, 2004; Hu and Liu, 2004; Popescu and Etzioni, 2005)) is the research in text most closely related to our work. Of the sentence-level research, the most similar is work by Raaijmakers and Kraaij (2008) comparing word-spanning character n-grams to word-internal character n-grams for subjectivity classification in news data. They found that character n-grams spanning words perform the best.

Research on recognizing subjective content in multiparty conversation includes work by Somasundaran et al. (2007) on recognizing sentiments and arguing in meetings, work by Neiberg el al. (2006) on recognizing positive, negative, and neutral emotions in meetings, work on recognizing agreements and disagreements in meetings (Hillard et al., 2003; Galley et al., 2004; Hahn et al., 2006), and work by Wrede and Shriberg (2003) on recognizing meeting hotspots. Somasundaran et al. use lexical and discourse features to recognize sentences and turns where meeting participants express sentiments or arguing. They also use the AMI corpus in their work; however, the use of different annotations and task definitions makes it impossible to directly compare their results and ours. Neiberg et al. use acoustic–prosodic features (Mel-frequency Cepstral Coefficients (MFCCs) and pitch features) and lexical n-grams for recognizing emotions in the ISL Meeting Corpus (Laskowski and Burger, 2006).

Agreements and disagreements are a subset of the private states represented by the positive and negative subjective categories used in this work. To recognise agreements and disagreements automatically, Hillard et al. train 3-way decision tree classifiers (agreement, disagreement, other) using both word-based and prosodic features. Galley et al. model this task as a sequence tagging problem, and investigate whether features capturing speaker interactions are useful for recognizing agreements and disagreements. Hahn et al. investigate the use of contrast classifiers (Peng et al., 2003) for the task, using only lexical features.

Hotspots are places in a meeting in which the participants are highly involved in the discussion. Although high involvement does not necessarily equate subjective content, in practice, we expect more sentiments, opinions, and arguments to be expressed when participants are highly involved in the discussion. In their work on recognizing meeting hotspots, Wrede and Shriberg focus on evaluating the contribution of various prosodic features, ignoring lexical features completely. The results of their study helped to inform our choice of prosodic features for the experiments in this paper.

7 Conclusions

In this paper, we investigated the use of prosodic features, word n-grams, character n-grams, and phoneme n-grams for subjectivity recognition and polarity classification of dialog acts in multiparty conversation. We show that character n-grams outperform prosodic features, word n-grams and phoneme n-grams in subjectiviy recognition and polarity classification. Combining these features significantly improves performance. Comparing the additive value of the four information sources available, prosodic information seem to be least informative while character-level information indeed proves to be a very valuable source. For subjectivity recognition, a combination of prosodic, word-level, character-level, and phoneme-level information yields the best performance. For polarity classification, the best performance is achieved with a combination of words, characters and phonemes.

References

R. Banse and K. R. Scherer. 1996. Acoustic profiles in vocal emotion expression. *Journal of Personality and Social Psychology*, pages 614–636.

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. In *Proceedings of the Measuring Behavior Symposium on "Annotating and Measuring Meeting Behavior"*.

J. Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20:37–46.

J. Eisner. 1996. An empirical comparison of probability models for dependency grammar. In *Technical Report IRCS-96-11, Institute for Research in Cognitive Science, University of Pennsylvania*.

M. Galley, K. 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 ACL*.

S. Hahn, R. Ladner, and M. Ostendorf. 2006. Agreement/disagreement classification: Exploiting unlabeled data using contrast classifiers. In *Proceedings of HLT/NAACL*.

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

P. Hsueh and J. Moore. 2006. Automatic topic segmentation and lablelling in multiparty dialogue. In *Proceedings of IEEE/ACM Workshop on Spoken Language Technology*.

P. Hsueh. 2008. Audio-based unsupervised segmentation of meeting dialogue. In *Proceedings of ICASSP*.

M. Hu and B. Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of KDD*.

F. Jelinek and R. L. Mercer. 1980. Interpolated estimation of Markov source parameters from sparse data. In *Proceedings, Workshop on Pattern Recognition in Practice*, pages 381–397.

I. Kanaris and E. Stamatatos. 2007. Webpage genre identification using variable-length character n-grams. In *Proceedings of ICTAI*.

V. Keselj, F. Peng, N. Cercone, and C. Thomas. 2003. N-gram-based author profiles for authorship attribution. In *Proceedings of PACLING*.

S. Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of Coling*.

D. Klein, J. Smarr, H. Nguyen, and C.D. Manning. 2003. Named entity recognition with character-level models. In *Proceedings of CoNLL*.

K. Laskowski and S. Burger. 2006. Annotation and analysis of emotionally relevant behavior in the ISL meeting corpus. In *Proceedings of LREC 2006*.

D. Neiberg, K. Elenius, and K. Laskowski. 2006. Emotion recognition in spontaneous speech using GMMs. In *Proceedings of INTERSPEECH*.

K. Peng, S. Vucetic, B. Han, H. Xie, and Z Obradovic. 2003. Exploiting unlabeled data for improving accuracy of predictive data mining. In *Proceedings of ICDM*.

A. Popescu and O. Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of HLT/EMNLP*.

M. Purver, P. Ehlen, and J. Niekrasz. 2006. Detecting action items in multi-party meetings: Annotation and initial experiments. In *Proceedings of MLMI*.

R. Quirk, S. Greenbaum, G. Leech, and J. Svartvik. 1985. *A Comprehensive Grammar of the English Language*. Longman, New York.

S. Raaijmakers and W. Kraaij. 2008. A shallow approach to subjectivity classification. In *Proceedings of ICWSM*.

S. Raaijmakers. 2007. Sentiment classification with interpolated information diffusion kernels. In *Proceedings of the First International Workshop on Data Mining and Audience Intelligence for Advertising (ADKDD'07)*.

E. Riloff and J. Wiebe. 2003. Learning extraction patterns for subjective expressions. In *Proceedings of EMNLP*.

R. E. Schapire and Y. Singer. 2000. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168.

S. Somasundaran, J. Ruppenhofer, and J. Wiebe. 2007. Detecting arguing and sentiment in meetings. In *Proceedings of SIGdial*.

E. Stamatatos. 2006. Ensemble-based author identification using character n-grams. In *Proceedings of TIR*.

T. Wilson. 2008. Annotating subjective content in meetings. In *Proceedings of LREC*.

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

H. Yu and V. Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of EMNLP*.

D. Zhang and W. S. Lee. 2006. Extracting key-substring-group features for text classification. In *Proceedings of KDD*.

Adapting a Lexicalized-Grammar Parser to Contrasting Domains

Laura Rimell and **Stephen Clark**
Oxford University Computing Laboratory
Wolfson Building, Parks Road
Oxford, OX1 3QD, UK
{laura.rimell,stephen.clark}@comlab.ox.ac.uk

Abstract

Most state-of-the-art wide-coverage parsers are trained on newspaper text and suffer a loss of accuracy in other domains, making parser adaptation a pressing issue. In this paper we demonstrate that a CCG parser can be adapted to two new domains, biomedical text and questions for a QA system, by using manually-annotated training data at the POS and lexical category levels only. This approach achieves parser accuracy comparable to that on newspaper data without the need for annotated parse trees in the new domain. We find that retraining at the lexical category level yields a larger performance increase for questions than for biomedical text and analyze the two datasets to investigate why different domains might behave differently for parser adaptation.

1 Introduction

Most state-of-the-art wide-coverage parsers are based on the Penn Treebank (Marcus et al., 1993), making such parsers highly tuned to newspaper text. A pressing question facing the parsing community is how to adapt these parsers to other domains, such as biomedical research papers and web pages. A related question is how to improve the performance of these parsers on constructions that are rare in the Penn Treebank, such as questions. Questions are particularly important since a question parser is a component in most Question Answering (QA) systems (Harabagiu et al., 2001).

In this paper we investigate parser adaptation in the context of lexicalized grammars, by using a parser based on Combinatory Categorial Grammar (CCG) (Steedman, 2000). A key property of CCG is that it is lexicalized, meaning that each word in a sentence is associated with an elementary syntactic structure. In the case of CCG this is a lexical category expressing subcategorization information. We exploit this property of CCG by performing manual annotation in the new domain, but only up to this level of representation, where the annotation can be carried out relatively quickly. Since CCG lexical categories are so expressive, many of the syntactic characteristics of a domain are captured at this level.

The two domains we consider are the biomedical domain and questions for a QA system. We use the term “domain” somewhat loosely here, since questions are best described as a particular set of syntactic constructions, rather than a set of documents about a particular topic. However, we consider question data to be interesting in the context of domain adaptation for the following reasons: 1) there are few examples in the Penn Treebank (PTB) and so PTB parsers typically perform poorly on them; 2) questions form a fairly homogeneous set with respect to the syntactic constructions employed, and it is an interesting question how easy it is to adapt a parser to such data; and 3) QA is becoming an important example of NLP technology, and question parsing is an important task for QA systems.

The CCG parser we use (Clark and Curran, 2007b) makes use of three levels of representation: one, a POS tag level based on the fairly coarse-grained POS tags in the Penn Treebank; two, a lexical category level based on the more fine-grained CCG lexical categories, which are assigned to words by a CCG supertagger; and three, a hierarchical level consisting of CCG derivations. A key idea in this paper, following a pilot study in Clark et al. (2004), is to perform manual annotation only at the first two levels. Since the lexical category level consists of sequences of tags, rather than hierarchical derivations, the annotation can be performed relatively quickly.

For the biomedical and question domains we manually annotated approximately 1,000 and 2,000 sentences, respectively, with CCG lexical categories. We also created a gold standard set of grammatical relations (GR) in the Stanford format (de Marneffe et al., 2006), using 500 of the questions. For the biomedical domain we used the BioInfer corpus (Pyysalo et al., 2007a), an existing gold-standard GR resource also in the Stanford format. We evaluated the parser on both lexical category assignment and recovery of GRs.

The results show that the domain adaptation approach used here is successful in two very different domains, achieving parsing accuracy comparable to state-of-the-art accuracy for newspaper text. The results also show, however, that the two domains have different profiles with regard to the levels of representation used by the parser. We find that simply retraining the POS tagger used by the parser leads to a large improvement in performance for the biomedical domain, and that retraining the CCG supertagger on the annotated biomedical data improves the performance further. For the question data, retraining just the POS tagger also improves parser performance, but retraining the supertagger has a much greater effect. We perform some analysis of the two datasets in order to explain the different behaviours with regard to porting the CCG parser.

2 The CCG Parser

The CCG parser is described in detail in Clark and Curran (2007b) and so we provide only a brief description. The stages in the CCG parsing pipeline are as follows. First, a maximum entropy POS tagger assigns a single POS tag to each word in a sentence. POS tags are fairly coarse-grained grammatical labels indicating part-of-speech; the Penn Treebank set, used here, contains approximately 50 labels.

Second, a maximum entropy supertagger assigns CCG lexical categories to the words in the sentence. Lexical categories can be thought of as fine-grained POS tags expressing subcategorization information, i.e. information about the argument frame of the word. There are 425 categories in the set used by the CCG parser. Supertagging was originally developed for Lexicalized Tree Adjoining Grammar (Bangalore and Joshi, 1999), but has been particularly successful for wide-coverage CCG parsing (Clark and Curran, 2007b). Rather than assign a single category to each word, the supertagger operates as a multi-tagger, sometimes assigning more than one category if the context is not sufficiently discriminating to suggest a single tag (Curran et al., 2006). Since the taggers have linear time complexity, the first two stages can be performed extremely quickly.

Finally, the parsing stage combines the lexical categories, using a small set of combinatory rules that are part of the grammar of CCG, and builds a packed chart representation containing all the derivations which can be built from the lexical categories. The Viterbi algorithm efficiently finds the highest scoring derivation from the packed chart, using a log-linear model to score the derivations. The grammar and training data for the newspaper version of the CCG parser are obtained from CCGbank (Hockenmaier and Steedman, 2007), a CCG version of the Penn Treebank.

The aspect of the pipeline which is most relevant to this paper is the supertagging phase. Figure 1 gives an example sentence from each target domain, with the CCG lexical category assigned to each word shown below the word, and the POS tag to the right. Note that the categories contain a significant amount of grammatical information, in particular subcategorization information. The verb *acts* in the biomedical sentence, for example, looks for a prepositional phrase (PP, *as a linkage protein*) to its right and a noun phrase (NP, *Talin*) to its left, with the resulting category a declarative sentence (S[dcl]).

Bangalore and Joshi (1999) refer to supertagging as *almost parsing*, because once the correct lexical categories have been assigned, the parser is left with much less work to do. The CCG supertagger is not able to assign a single category to each word with extremely high accuracy — hence the need for it to operate as a multi-tagger — but even in multi-tagger mode it dramatically reduces the ambiguity passed through to the parser (Clark and Curran, 2007b).

$$\begin{array}{cccccccc} \text{Talin}|\text{NN} & \text{perhaps}|\text{RB} & \text{acts}|\text{VBZ} & \text{as}|\text{IN} & \text{a}|\text{DT} & \text{linkage}|\text{NN} & \text{protein}|\text{NN} & .|. \\ \overline{NP} & \overline{(S\backslash NP)/(S\backslash NP)} & \overline{(S[dcl]\backslash NP)/PP} & \overline{PP/NP} & \overline{NP[nb]/N} & \overline{N/N} & \overline{N} & \overline{.} \end{array}$$

$$\begin{array}{ccccccc} \text{What}|\text{WDT} & \text{king}|\text{NN} & \text{signed}|\text{VBD} & \text{the}|\text{DT} & \text{Magna}|\text{NNP} & \text{Carta}|\text{NNP} & ?|. \\ \overline{(S[wq]/(S[dcl]\backslash NP))/N} & \overline{N} & \overline{(S[dcl]\backslash NP)/NP} & \overline{NP[nb]/N} & \overline{N/N} & \overline{N} & \overline{.} \end{array}$$

Figure 1: Example sentences with lexical category assignment.

The parser has been evaluated on DepBank (King et al., 2003), using the GR scheme of Briscoe et al. (2006), and it scores 82.4% labelled precision and 81.2% labelled recall overall (Clark and Curran, 2007a). Section 4.4 describes how the CCG dependencies can be mapped into the Stanford GR scheme (de Marneffe et al., 2006) and gives the results of evaluating the parser on biomedical and question GR resources.

The CCG parser is particularly well suited to the biomedical and question domains. First, use of CCG allows recovery of long-distance dependencies. In the sentence *What does target heart rate mean?*, the word *What* is an underlying object of the verb *mean*. The parser recovers this information despite the distance between the two words. This capability is crucial for question parsing, and also useful in the biomedical field for extraction of relationships between biological entities. Additionally, the speed of the parser (tens of sentences per second) is useful for the large volumes of biomedical data that require processing for biomedical text mining.

3 Approach

Our approach to domain adaptation is to target the coarser-grained, less syntactically complex, levels of representation used by the parser, and to train new models with manually annotated data at these levels. The motivation for this approach is twofold. First, accuracy at each stage of the pipeline depends on accuracy at the earlier stages. If the POS tagger assigns incorrect tags, it is unlikely that the supertagger will be able to recover and produce the correct lexical categories, since it relies heavily on POS tags as features. Without the correct categories, the parser in turn will be unable to find a correct parse.

In the sentence *What year did the Vietnam War end?*, the newspaper-trained POS tagger incorrectly assigns the POS tag NN (common noun) to the verb *end*, since verb-final sentences are atypical for the PTB. As a result, the supertagger is virtually certain (greater than 99% probability) that the correct CCG lexical category for *end* is N (noun). The parser then assigns *the Vietnam War end* the structure of a noun phrase, and chooses an unusual subcategorization frame for *did* in which it takes three arguments: *What*, *year*, and *the Vietnam War end*.

In the sentence *How many siblings does she have?*, on the other hand, the supertagger assigns an incorrect category to the word *How* despite it having the correct POS tag (WRB for wh-adverb). The correct category is $((S[wq]/(S[q]/NP))/N)/(NP/N)$, which takes *many* (category NP/N) and *siblings* (category N) as arguments. Instead it is tagged as $S[wq]/S[q]$, the category for a sentential adverb (i.e. the manner reading of *how*), which prevents a correct parse. Our intention was that creating new training data at the lower levels of representation would improve the accuracy of the POS tagger and supertagger in the target domains, thereby improving the accuracy of later stages in the pipeline as well.

The second motivation for our approach is to reduce annotation overhead. Full syntactic derivations are costly to produce by hand. POS tags, however, are relatively easy to annotate; even an out-of-domain tagger will provide a good starting point, and manual correction is quick, especially in a domain without much unfamiliar vocabulary. CCG lexical categories require more expertise, but our experience shows that an out-of-domain supertagger can again provide a starting point for correction, and since the annotation is flat rather than hierarchical, we hypothesize that it is not as difficult or time-consuming as annotation of full derivations.

Our adaptation approach has been partially explored in previous work which targets one or another of the different levels of representation.

Lease and Charniak (2005) obtained an improvement in the accuracy of the Charniak (2000) parser, as well as POS tagging accuracy, when applied to the biomedical domain, by training a new POS tagger model with a combination of newspaper and biomedical data. The parser improvement was due solely to the new POS tagger, without retraining the parser model. Since the Charniak parser does not use a lexicalized grammar with an intermediate level of representation, any further improvements would have to come from the parser model itself.

Clark et al. (2004) obtained an improvement in CCG supertagging accuracy for *What*-questions by training a new supertagger model with a combination of newspaper and question data annotated with CCG lexical categories. Because a question resource annotated with GRs was not available, they did not perform a parser evaluation, and the effects of the POS tagging level were not compared to the lexical category level. In this paper, we extend the pilot experiments performed by Clark et al. (2004) in four ways. First, we use a larger corpus of TREC questions covering additional question types, thus extending the experiments to the question domain more broadly, as well as to the biomedical domain. Second, we create a gold standard GR resource enabling a full parser evaluation on question data. Third, we show that the POS level is important for adaptation, reinforcing the work of Lease and Charniak (2005). A key finding of the present paper is that the combination of retraining at the POS tag and lexical category levels provides additional improvements beyond those gained by retraining at a single level. Finally, we provide analysis comparing the adaptation methodology for question and biomedical data.

Hara et al. (2007) followed a similar approach to Clark et al. (2004), using the parser of Ninomiya et al. (2006), a version of the Enju parser (Miyao and Tsujii, 2005). Enju is based on HPSG, a lexicalized grammar formalism. They obtained an improvement in parsing accuracy in the biomedical domain by training a new probabilistic model of lexical entry assignments on a combination of newspaper and biomedical data without changing the original newspaper-trained parsing model. Hara et al. (2007) did not consider the role of POS tagging. The lexical category data in Hara et al. (2007) was derived from a gold standard treebank, while the annotation of lexical categories in this paper was performed without reference to gold standard syntactic derivations.

Judge et al. (2006) produced a corpus of 4,000 questions annotated with syntactic trees, and obtained an improvement in parsing accuracy for Bikel's reimplementation of the Collins parser (Collins, 1997) by training a new parser model with a combination of newspaper and question data. Our approach differs in retraining only at the levels of representation below parse trees.

4 Experiments and Results

4.1 Resources

We have used a combination of existing resources and new, manually annotated data. The baseline POS tagger, supertagger, and parser are trained on WSJ Sections 02-21 of CCGbank. The baseline performance at each level of representation is on WSJ Section 00 of CCGbank, which contains 1913 sentences and approximately 45,000 words.

For the biomedical domain, we trained the POS tagger on gold-standard POS tags from GENIA (Kim et al., 2003), a corpus of 2,000 MEDLINE abstracts containing a total of approximately 18,500 sentences and 440,000 words. We also annotated the first 1,000 sentences of GENIA with CCG lexical categories. This set of 1,000 sentences, containing approximately 27,000 words, was used for POS tagger evaluation and for development and evaluation of a new supertagger model. For parser evaluation, we used BioInfer (Pyysalo et al., 2007a), a corpus of MEDLINE abstracts (on a different topic from those in GENIA) containing 1,100 sentences, and with syntactic dependencies encoded as grammatical relations in the Stanford GR format. We used the same evaluation set of 500 sentences as in Pyysalo et al. (2007b), and the remaining 600 for development of the mapping to Stanford format. Two parsers have already been evaluated on BioInfer, which makes it a useful resource for comparative evaluation.

For the question domain, we extended the dataset described in Clark et al. (2004). That dataset contained 1,171 questions beginning with the word *What*, from the TREC 9-12 competitions (2000-2003), manually POS tagged and annotated with

CCG lexical categories. We annotated all the additional TREC question types and improved the existing annotation, for a total of 1,828 sentences. We additionally annotated a random subset of 500 of these with GRs in the Stanford format. This subset served as our evaluation set at all levels of representation. It contains approximately 4,000 words, fewer than the other domains because of the significantly shorter sentence lengths of typical questions. The remaining 1,328 sentences were used as training data. A set of about a dozen sentences from the evaluation and training sets were used to develop the mapping to Stanford format for lexical categories not occurring in the biomedical data.

4.2 POS tagger

We began by training new models at the POS tag level of representation. All datasets use the PTB tagset. As a baseline, we used the original WSJ 02-21 model on the biomedical and question datasets. For comparison we also evaluated on Section 00 using the WSJ-trained model.

For the question data, the new POS tagger was trained on CCGbank Sections 02-21 plus ten copies of the 1,328 training sentences. The WSJ data provides additional robustness and wide grammatical coverage, and the weighting factor of ten was chosen in preliminary experiments to prevent the newspaper data from "overwhelming" the question data. For the biomedical data, the new POS tagger was trained on the full GENIA corpus, minus the first 1,000 sentences. GENIA is large enough that combination with the newspaper data was not needed.

Table 1 gives the results. For both of the new domains the performance of the WSJ model decreased compared to Section 00, but the retrained model performed at least as well as the WSJ model did on 00.[1] Improving the POS tagger performance has a positive effect on the performance of the supertagger and parser, which will be discussed in Sections 4.3-4.4.

[1]Since GENIA does not use the proper noun tag, NNP, for names of genes and other biomedical entities, all figures in this paper collapse the NNP-NN distinction where relevant for biomedical data. The question data uses NNP and the distinction is not collapsed.

	WSJ 02-21	Retrained
Sec. 00	96.7	—
Qus	92.2	97.1
Bio	93.4	98.7

Table 1: POS tagger accuracy (%) for original and retrained models.

	Orig pipeline	Retrained POS	Retrained POS and super
Sec. 00	91.5	—	—
Qus	71.6	74.0	92.1
Bio	89.0	91.2	93.0

Table 2: Supertagging accuracy (%) and the effect of retraining the POS model and the supertagger model.

4.3 Supertagger

We next trained new models at the CCG lexical category level. The training data consisted of manually annotated biomedical and question sentences; specifically, lexical categories were automatically assigned by the original parsing pipeline and then manually corrected. Whenever possible we used categories from the parser's original set of 425, although occasionally it was necessary to use a new category for a syntactic construction not occurring in CCGbank Sections 02-21. (The parser can be configured to recognize additional categories.) Question data in particular requires the use of categories that are rare or unseen in CCGbank.

For the questions, the new supertagger model, like the POS tagger, was trained on WSJ 02-21 plus ten copies of the 1,328 training sentences. For the biomedical data, a ten-fold cross-validation was performed, training each supertagger model on WSJ 02-21 plus ten copies of 90% of the 1,000 annotated sentences. Table 2 gives the supertagger accuracy with and without the retrained POS and supertagger models. The figure for the retrained biomedical supertagger is the average of the ten-fold split.

The results show an improvement in accuracy of lexical category assignment solely from retraining the POS tagger, and an additional improvement from retraining the supertagger. Supertagger accuracy for the two domains with a retrained supertagger was comparable, and in both cases was at least as high

What car company invented the Edsel?
(nsubj invented company)
(det Edsel the)
(dobj invented Edsel)
(det company What)
(nn company car)

Figure 2: Example of grammatical relations in the Stanford grammatical relation format.

as for the original pipeline on Section 00. The question data started from a much lower baseline figure, however.

4.4 Parser

We evaluated the parser on the 500 questions annotated with Stanford GRs and on the 500 evaluation sentences from the BioInfer corpus. We used the original newspaper pipeline, a pipeline with a retrained POS tagger, and a pipeline with both a retrained POS tagger and supertagger.

In order to perform these evaluations we develooped a mapping from the parser's native CCG syntactic dependencies to GRs in the Stanford format. The mapping was based on the same principles as the mapping that produces GR output in the style of Briscoe et al. (2006). These principles are discussed in detail in Clark and Curran (2007a); in summary, the argument slots in the CCG dependencies are mapped to argument slots in Stanford GRs, a fairly complex, many-to-many mapping. An additional post-processing script applies some manually developed rules to bring the output closer to the Stanford format. Figure 2 gives an example of Stanford GRs, where the label of the relation is followed by two arguments, head and dependent.

Table 3 gives the results of the parser evaluation on GRs. Since the parser model was not retrained, the improvements in accuracy are due solely to the new POS and supertaggers. The results are given as an F-score over labelled GRs.[2]

The F-scores given in Table 3 are only for sentences for which a parse was found. However, there were also improvements in coverage with the retrained models. For the question data, parser coverage was 94% for the original pipeline and the pipeline with just the retrained POS tagger, and 99.6% with the retrained POS and supertaggers. For the biomedical data, coverage was 97.2% for the original pipeline, 99.0% for the pipeline with the retrained POS tagger, and 99.8% for the pipeline with the retrained POS and supertaggers.

[2]Only GRs at the lowest level of the Stanford hierarchy were considered in the evaluation; more generic relations such as *dependent* were not considered.

	Orig POS and super	New POS	New POS and super
Qus	64.4	69.4	86.6
BioInfer	76.0	80.4	81.5

Table 3: Parser F-score on grammatical relations and the effect of retraining the POS and supertagger models.

The final accuracy for both domains is in the same range as that of the original parser on newspaper data (81.8%) (Clark and Curran, 2007b), although the results are not directly comparable, since the newspaper resource uses a different GR scheme. For the BioInfer corpus, the final accuracy is also in line with results reported in the literature for other parsers (Pyysalo et al., 2007b). (No comparable GR results are available for questions.) A score in this range is thought to be near the upper bound when evaluating a CCG parser on GRs, since some loss is inherent in the mapping to GRs (Clark and Curran, 2007a).

5 Analysis

Although domain adaptation was successful for both of our target domains, the impact of the different levels of representation on parsing accuracy was not uniform. Table 3 shows that retraining the POS tagger accounted for a greater proportion of the improvement on biomedical data, while retraining the supertagger accounted for a much greater proportion on questions. In this section we discuss some of the differences between the domains which may have contributed to their behaviour in this regard, with the intention of highlighting attributes that may be relevant for domain adaptation in general.

Informally, we believe that the main difference between newspaper and biomedical text is vocabulary, and that their syntactic structures are essentially similar (with some isolated exceptions, such as more frequent use of parentheses and comma-separated

	Tag	Errors	Freq confused
Qus	WDT	129	WP
	VB	46	NN, VBP
	NNP	33	JJ, NN
	NN	32	JJ, NNP
Bio	NN	801	JJ, CD
	JJ	268	NN, VBN
	VBN	113	JJ, VBD
	FW	95	NN, IN

Table 4: Tags with the most frequent errors by the newspaper-trained POS tagger and the tags they were most frequently confused with.

lists in biomedical text). Once the POS tagger had been retrained for biomedical text, accounting for unfamiliar vocabulary, the original supertagger already performed well. The main difference between newspaper and question data, on the other hand, is syntactic. Retraining the POS tagger for questions therefore had less effect; even with the correct POS tags the supertagger was unable to assign the correct lexical categories. Since lexical categories encode syntactic information, the domain with the more divergent syntax is likely to benefit most from new training data at the lexical category level.

5.1 POS tagger

Table 1 showed that the accuracy of the newspaper-trained POS tagger was in the same range for both biomedical and question data. However, the distribution of errors was different. Table 4 shows the tags with the most frequent errors, accounting for about 75% of all POS tag errors in each domain, and the tags that they were most frequently confused with.

For the question data, the most frequent error was tagging a wh-determiner (WDT) as a wh-pronoun (WP). A determiner combines with a noun to form a noun phrase, as in the sentence <u>*What*</u> *Liverpool club spawned the Beatles?*. A pronoun, on the other hand, is a noun phrase in its own right, as in <u>*What*</u> *are the colors of the German flag?*. This tagger error arises from the fact that the word *What* occurs only once in WSJ 02-21 with a WDT tag. The second most common error was on bare verbs (VB), because the newspaper model gives a low probability of bare verbs occurring in sentence-final position, or not directly following an auxiliary.

	Unknown word rate	Unknown word-POS rate
Sec. 00	3.8	4.4
Qus	7.5	8.3
Bio	23.6	25.3

Table 5: Unknown word rate and word-POS tag pair rate (%) compared to WSJ 02-21 (by token).

For the biomedical data, the most frequent errors by far were confusions of noun (NN) and adjective (JJ). This is most likely due to the prevalence of long noun phrases in the biomedical data, such as *major histocompatibility complex class II molecules*. Although the words preceding the head noun are recognized as nominal modifiers, the classification into noun and adjective is difficult, especially when the word is previously unseen. There were also problems distinguishing verbal past participles (VBN) from adjectives (JJ) and identifying foreign words (FW), for example the phrase *in vitro*.

The fact that the newspaper-trained POS tagger performed comparably in the two target domains (Table 1) is surprising, since their lexical profiles are quite different. Lease and Charniak (2005) discussed unknown word rate as a predictor of POS tagger accuracy. However, the unknown word rate compared with WSJ 02-21 is much higher for the biomedical data than for the question data, as seen in Table 5. (The unknown word rate for the question data is still higher than that for WSJ 00, which may be due to the high proportion of proper nouns in the question data.)

Some POS tagging errors can be attributed, not to an unknown word, but to the use of a known word with an unfamiliar tag (as in the WDT example above). However, it is not the case that the question data contains many known words with unknown tags, since the rate of unknown word-tag pairs is also much higher for biomedical than for question data, as seen in the rightmost column of Table 5.

We do know that the newspaper-trained POS tagger performs better on unknown words for biomedical (84.7%) than for question data (80.4%). We hypothesize that the syntactic context of the biomedical data, being more similar to newspaper data, provides more information for the POS tagger in

	WSJ 02-21	New training sets
3-grams		
Sec. 00	0.4	—
Qus	3.6	0.7
Bio	0.7	0.5
5-grams		
Sec. 00	12.1	—
Qus	22.0	7.4
Bio	10.9	9.2

Table 6: Unknown POS n-gram rate (%) compared to WSJ 02-21, and when in-domain data is added (by token).

biomedical than in question data. Syntactic differences are discussed in the next section.

5.2 Supertagger

To quantify the syntactic distance between domains, we propose using the unknown POS n-gram rate compared to WSJ Sections 02-21. In the absence of parse trees, POS n-grams can serve as a rough proxy for the syntactic characteristics of a domain, reflecting local word order configurations. POS n-grams have been used in document modeling for text categorization (Baayen et al., 1996; Argamon-Engelson et al., 1998), but we believe our proposed use of the unknown POS n-gram rate is novel.

The leftmost column of Table 6 gives the unknown POS trigram and 5-gram rates compared to WSJ Sections 02-21. The rates for the biomedical data are quite similar to those for Section 00. The question data, however, shows higher rates of unknown POS n-grams.

For both biomedical and question data, adding in-domain data to the training set makes its syntactic profile more like that of the evaluation set. The rightmost column of Table 6 shows the unknown POS n-gram rates compared to the datasets used for training the new supertagger models, consisting of WSJ 02-21 plus annotated question or biomedical data. (For the biomedical data, the figures are averages of the same ten-fold split used for evaluation). It can be seen that adding in-domain data reduces the rate of unknown POS n-grams to about the same level observed for newspaper text.

The unknown POS n-gram rate requires POS tagged data for a new domain and thus cannot be used with unlabelled data. However, since POS tagging is relatively inexpensive, it might be possible to use this rate as one measure of syntactic distance between a training corpus and a target domain, prior to undertaking parser domain adaptation. The measure does not capture all aspects of syntactic distance, however. As pointed out by an anonymous reviewer, if the syntactic tree structures are similar across domains but lexical distributions are different – e.g. a large number of words with unfamiliar categories in the new domain – this measure will not be sensitive to the difference.

Another useful measure for comparing domain adaptation in the biomedical and question domains is frequent POS n-grams. Table 7 shows how many of the 20 most frequent POS n-grams in each dataset overlap with the 20 most frequent POS n-grams in WSJ 02-21. It can be seen that the overlap is the highest for Section 00, but much lower for the question data than for the biomedical data, again demonstrating that the question data makes frequent use of syntactic constructions which are rare in the PTB.

Table 8 shows the four most frequent POS trigrams in WSJ Sections 02-21,[3] and the four most frequent POS trigrams in the biomedical and question data that are not among the 20 most frequent

	3-grams	5-grams
Sec. 00	18	19
Qus	8	5
Bio	16	13

Table 7: Number of the 20 most frequent POS n-grams that are also in the 20 most frequent POS n-grams of WSJ Sections 02-21.

WSJ 02-21	Bio	Qus
. — —	JJ NN NN	— — WP
IN DT NN	IN JJ NN	— WP VBZ
NN . —	NN IN JJ	— — WDT
DT JJ NN	NNS IN NN	WP VBZ DT

Table 8: Four most frequent POS trigrams for WSJ 02-21; four most frequent POS trigrams for biomedical and question data that are not in the 20 most frequent for WSJ 02-21. The dash represents the sentence boundary.

[3]Collapsing the NNP-NN distinction yields a slightly different set.

for WSJ 02-21. The frequent question trigrams include two sentence-initial question words as well as the pattern — WP VBZ, occurring in sentences beginning with e.g. *What is* or *Who is*. Though not among the top four, the pattern VB . —, representing a sentence-final bare verb, is also frequent. The most frequent biomedical POS trigrams are not dramatically different from the newspaper trigrams, but do appear to reflect the prevalence of NPs and PPs in the data.

One final measure of syntactic distance is the frequency with which CCG lexical categories that are rare or unseen in CCGbank are used in a domain. It is typical to use a few such categories, even for in-domain data, for unusual syntactic constructions, but each one is usually used only a handful of times. The question data is unique in the frequency with which previously rare or unseen categories are required. For example, the unseen category $(S[wq]/S[q])/N$, representing the word *What* in a question such as *What day did Nintendo 64 come out?* is used 11 times in the evaluation set; the rare category $(S[wq]/(S[dcl]\backslash NP))/N$, used in subject questions like *Which river runs through Dublin?*, is used 61 times; and the rare category $(S[q]/(S[pss]\backslash NP))/NP$, representing passive verbs in sentences like *What is Jane Goodall known for?*, is used 59 times.

6 Conclusion

We have targeted lower levels of representation in order to adapt a lexicalized-grammar parser to two new domains, biomedical text and questions. Although each of the lower levels has been targeted independently in previous work, this is the first study that examines both levels together to determine how they affect parsing accuracy. We achieved an accuracy on grammatical relations in the same range as that of the original parser for newspaper text, without requiring costly annotation of full parse trees.

Both biomedical and question data are domains in which there is an immediate need for accurate parsing. The question dataset is in some ways an extreme example for domain adaptation, since the sentences are syntactically uniform; on the other hand, it is of interest as a set of constructions where the parser initially performed poorly, and is a realistic parsing challenge in the context of QA systems.

Interestingly, although an increase in accuracy at each stage of the pipeline did yield an increase at the following stage, these increases were not uniform across the two domains. The new POS tagger model was responsible for most of the improvement in parsing for the biomedical domain, while the new supertagger model was necessary to see a large improvement in the question domain. We attribute this to the fact that question syntax is significantly different from newspaper syntax. We expect these considerations to apply to any lexicalized-grammar parser.

Of course, it would be useful to have a way of predicting which level of annotation would be most effective for adapting to a new domain before the annotation begins. The utility of measures such as unknown word rate (which can be performed with unlabelled data) and unknown POS n-gram rate (which can be performed with only POS tags) is not yet sufficiently clear to rely on them as predictive measures, but it seems a fruitful avenue for future work to investigate the importance of such measures for parser domain adaptation.

Acknowledgments

We would like to thank Marie-Catherine de Marneffe for advice on the use of the Stanford GR format, Sampo Pyysalo for sharing information about the BioInfer corpus, and Mark Steedman for advice on encoding question data in CCG. We would also like to thank three anonymous reviewers for their suggestions. This work was supported by EPSRC grant EP/E035698/1: Accurate and Efficient Parsing of Biomedical Text.

References

Shlomo Argamon-Engelson, Moshe Koppel, and Galit Avneri. 1998. Style-based text categorization: What newspaper am I reading? In *Proceedings of AAAI Workshop on Learning for Text Categorization*, pages 1–4.

Harald Baayen, Hans Van Halteren, and Fiona Tweedie. 1996. Outside the cave of shadows: Using syntactic annotation to enhance authorship attribution. *Literary and Linguistic Computing*, 11(3):121–131.

Srinivas Bangalore and Aravind Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the Interactive Demo Session of the Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING/ACL-06)*, Sydney, Austrailia.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Meeting of the NAACL*, pages 132–139, Seattle, WA.

Stephen Clark and James R. Curran. 2007a. Formalism-independent parser evaluation with CCG and DepBank. In *Proceedings of the 45th Meeting of the ACL*, pages 248–255, Prague, Czech Republic.

Stephen Clark and James R. Curran. 2007b. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Stephen Clark, Mark Steedman, and James R. Curran. 2004. Object-extraction and question-parsing using CCG. In *Proceedings of the EMNLP Conference*, pages 111–118, Barcelona, Spain.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Meeting of the ACL*, pages 16–23, Madrid, Spain.

James R. Curran, Stephen Clark, and David Vadas. 2006. Multi-tagging for lexicalized-grammar parsing. In *Proceedings of the Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING/ACL-06)*, pages 697–704, Sydney, Austrailia.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th LREC Conference*, pages 449–454, Genoa, Italy.

Tadayoshi Hara, Yusuke Miyao, and Jun'ichi Tsujii. 2007. Evaluating impact of re-training a lexical disambiguation model on domain adaptation of an HPSG parser. In *Proceedings of IWPT*, pages 11–22, Prague, Czech Republic.

Sanda Harabagiu, Dan Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan Bunescu, Roxana Girju, Vasile Rus, and Paul Morarescu. 2001. The role of lexico-semantic feedback in open-domain textual question-answering. In *Proceedings of the 39th Meeting of the ACL*, pages 274–281, Toulose, France.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

John Judge, Aoife Cahill, and Josef van Genabith. 2006. Questionbank: Creating a corpus of parse-annotated questions. In *Proceedings of the Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics*, pages 497–504, Sydney, Australia.

Jin-Dong Kim, Tomoko Ohta, Yuka Teteisi, and Jun'ichi Tsujii. 2003. GENIA corpus – a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19:i180–i182.

Tracy H. King, Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M. Kaplan. 2003. The PARC 700 Dependency Bank. In *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora*, Budapest, Hungary.

Matthew Lease and Eugene Charniak. 2005. Parsing biomedical literature. In *Proceedings of the Second International Joint Conference on Natural Language Processing (IJCNLP-05)*, Jeju Island, Korea.

Mitchell Marcus, Beatrice Santorini, and Mary Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Yusuke Miyao and Jun'ichi Tsujii. 2005. Probabilistic disambiguation models for wide-coverage HPSG parsing. In *Proceedings of the 43rd meeting of the ACL*, pages 83–90, University of Michigan, Ann Arbor.

Takashi Ninomiya, Takuya Matsuzaki, Yoshimasa Tsuruoka, Yusuke Miyao, and Jun'ichi Tsujii. 2006. Extremely lexicalized models for accurate and fast HPSG parsing. In *Proceedings of the EMNLP Conference*.

Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Björne, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. 2007a. BioInfer: A corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, 8:50.

Sampo Pyysalo, Filip Ginter, Veronika Laippala, Katri Haverinen, Juho Heimonen, and Tapio Salakoski. 2007b. On the unification of syntactic annotations under the stanford dependency scheme: A case study on BioInfer and GENIA. In *ACL'07 workshop on Biological, translational, and clinical language processing*, pages 25–32, Prague, Czech Republic.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, MA.

Improving Interactive Machine Translation via Mouse Actions

Germán Sanchis-Trilles and **Daniel Ortiz-Martínez** and **Jorge Civera**
Instituto Tecnológico de Informática
Universidad Politécnica de Valencia
{gsanchis,dortiz,jorcisai}@iti.upv.es

Francisco Casacuberta and **Enrique Vidal**
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
{fcn,evidal}@dsic.upv.es

Hieu Hoang
University of Edinburgh
hhoang@sms.ed.ac.uk

Abstract

Although Machine Translation (MT) is a very active research field which is receiving an increasing amount of attention from the research community, the results that current MT systems are capable of producing are still quite far away from perfection. Because of this, and in order to build systems that yield correct translations, human knowledge must be integrated into the translation process, which will be carried out in our case in an Interactive-Predictive (IP) framework. In this paper, we show that considering Mouse Actions as a significant information source for the underlying system improves the productivity of the human translator involved. In addition, we also show that the initial translations that the MT system provides can be quickly improved by an expert by only performing additional Mouse Actions. In this work, we will be using word graphs as an efficient interface between a phrase-based MT system and the IP engine.

1 Introduction

Information technology advances in modern society have led to the need of more efficient methods of translation. It is important to remark that current MT systems are not able to produce ready-to-use texts (Kay, 1997; Hutchins, 1999; Arnold, 2003). Indeed, MT systems are usually limited to specific semantic domains and the translations provided require human post-editing in order to achieve a correct high-quality translation.

A way of taking advantage of MT systems is to combine them with the knowledge of a human translator, constituting the so-called Computer-Assisted Translation (CAT) paradigm. CAT offers different approaches in order to benefit from the synergy between humans and MT systems.

An important contribution to interactive CAT technology was carried out around the TransType (TT) project (Langlais et al., 2002; Foster et al., 2002; Foster, 2002; Och et al., 2003). This project entailed an interesting focus shift in which interaction directly aimed at the production of the target text, rather than at the disambiguation of the source text, as in former interactive systems. The idea proposed was to embed data driven MT techniques within the interactive translation environment.

Following these TT ideas, (Barrachina and others, 2008) propose the usage of fully-fledged statistical MT (SMT) systems to produce full target sentence hypotheses, or portions thereof, which can be partially or completely accepted and amended by a human translator. Each partial correct text segment is then used by the SMT system as additional information to achieve further, hopefully improved suggestions. In this paper, we also focus on the interactive and predictive, statistical MT (IMT) approach to CAT. The IMT paradigm fits well within the *Interactive Pattern Recognition* framework introduced in (Vidal and others, 2007).

SOURCE $(\mathbf{x})$**:** Para encender la impresora:
REFERENCE $(\mathbf{y})$**:** To power on the printer:

ITER-0	**(p)**	( )
	$(\hat{\mathbf{s}}_h)$	*To switch on:*
ITER-1	**(p)**	To
	$(\mathbf{s}_l)$	*switch on:*
	(k)	power
	$(\hat{\mathbf{s}}_h)$	*on the printer:*
ITER-2	**(p)**	To power on the printer:
	$(\mathbf{s}_l)$	( )
	(k)	(#)
	$(\hat{\mathbf{s}}_h)$	( )
FINAL	$(\mathbf{p} \equiv \mathbf{y})$	To power on the printer:

Figure 1: IMT session to translate a Spanish sentence into English. Non-validated hypotheses are displayed in italics, whereas accepted prefixes are printed in normal font.

Figure 1 illustrates a typical IMT session. Initially, the user is given an input sentence $\mathbf{x}$ to be translated. The reference $\mathbf{y}$ provided is the translation that the user would like to achieve at the end of the IMT session. At iteration 0, the user does not supply any correct text prefix to the system, for this reason $\mathbf{p}$ is shown as empty. Therefore, the IMT system has to provide an initial complete translation $\mathbf{s}_h$, as it were a conventional SMT system. At the next iteration, the user validates a prefix $\mathbf{p}$ as correct by positioning the cursor in a certain position of $\mathbf{s}_h$. In this case, after the words "*To print a*". Implicitly, he is also marking the rest of the sentence, the suffix $\mathbf{s}_l$, as potentially incorrect. Next, he introduces a new word k, which is assumed to be different from the first word s_{l_1} in the suffix $\mathbf{s}_l$ which was not validated, $k \neq s_{l_1}$. This being done, the system suggests a new suffix hypothesis $\hat{\mathbf{s}}_h$, subject to $\hat{s}_{h_1} = k$. Again, the user validates a new prefix, introduces a new word and so forth. The process continues until the whole sentence is correct that is validated introducing the special word "#".

As the reader could devise from the IMT session described above, IMT aims at reducing the effort and increasing the productivity of translators, while preserving high-quality translation. For instance, in Figure 1, only three interactions were necessary in order to achieve the reference translation.

In this paper, we will show how Mouse Actions performed by the human expert can be taken advantage of in order to further reduce this effort.

2 Statistical interactive-predictive MT

In this section we will briefly describe the statistical framework of IMT. IMT can be seen as an evolution of the SMT framework, which has proved to be an efficient framework for building state-of-the-art MT systems with little human effort, whenever adequate corpora are available (Hutchings and Somers, 1992). The fundamental equation of the statistical approach to MT is

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\text{argmax}}\, Pr(\mathbf{y} \mid \mathbf{x}) \tag{1}$$

$$= \underset{\mathbf{y}}{\text{argmax}}\, Pr(\mathbf{x} \mid \mathbf{y})\, Pr(\mathbf{y}) \tag{2}$$

where $Pr(\mathbf{x} \mid \mathbf{y})$ is the *translation model* modelling the correlation between source and target sentence and $Pr(\mathbf{y})$ is the *language model* representing the well-formedness of the candidate translation $\mathbf{y}$.

In practise, the direct modelling of the posterior probability $Pr(\mathbf{y}|\mathbf{x})$ has been widely adopted. To this purpose, different authors (Papineni et al., 1998; Och and Ney, 2002) propose the use of the so-called log-linear models, where the decision rule is given by the expression

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\text{argmax}} \sum_{m=1}^{M} \lambda_m h_m(\mathbf{x}, \mathbf{y}) \tag{3}$$

where $h_m(\mathbf{x}, \mathbf{y})$ is a score function representing an important feature for the translation of $\mathbf{x}$ into $\mathbf{y}$, M is the number of models (or features) and λ_m are the weights of the log-linear combination.

One of the most popular instantiations of log-linear models is that including phrase-based (PB) models (Zens et al., 2002; Koehn et al., 2003). Phrase-based models allow to capture contextual information to learn translations for whole phrases instead of single words. The basic idea of phrase-based translation is to segment the source sentence into phrases, then to translate each source phrase into a target phrase, and finally to reorder the translated target phrases in order to compose the target sentence. Phrase-based models were employed throughout this work.

In log-linear models, the maximisation problem stated in Eq. 3 is solved by means of the beam search algorithm[1] which was initially introduced in (Lowerre, 1976) for its application in the field of speech recognition. The beam search algorithm attempts to generate partial solutions, called *hypotheses*, until a complete sentence is found; these hypotheses are stored in a stack and ordered by their *score*. Such a score is given by the log-linear combination of feature functions.

However, Eq. 1 needs to be modified according to the IMT scenario in order to take into account part of the target sentence that is already translated, that is $\mathbf{p}$ and k

$$\hat{\mathbf{s}}_h = \underset{\mathbf{s}_h}{\operatorname{argmax}}\, Pr(\mathbf{s}_h|\mathbf{x},\mathbf{p},k) \tag{4}$$

where the maximisation problem is defined over the suffix $\mathbf{s}_h$. This allows us to rewrite Eq. 4, by decomposing the right side appropriately and eliminating constant terms, achieving the equivalent criterion

$$\hat{\mathbf{s}}_h = \underset{\mathbf{s}_h}{\operatorname{argmax}}\, Pr(\mathbf{p},k,\mathbf{s}_h|\mathbf{x}). \tag{5}$$

An example of the intuition behind these variables can be seen in Figure 1.

Note that, since $(\mathbf{p}\,k\,\mathbf{s}_h) = \mathbf{y}$, Eq. 5 is very similar to Eq. 1. The main difference is that the argmax search is now performed over the set of suffixes $\mathbf{s}_h$ that complete $(\mathbf{p}\,k)$ instead of complete sentences ($\mathbf{y}$ in Eq. 1). This implies that we can use the same models if the search procedures are adequately modified (Barrachina and others, 2008).

[1]Also known as stack decoding algorithm.

3 Phrase-based IMT

The phrase-based approach presented above can be easily adapted for its use in an IMT scenario. The most important modification is to rely on a word graph that represents possible translations of the given source sentence. The use of word graphs in IMT has been studied in (Barrachina and others, 2008) in combination with two different translation techniques, namely, the Alignment Templates technique (Och et al., 1999; Och and Ney, 2004), and the Stochastic Finite State Transducers technique (Casacuberta and Vidal, 2007).

3.1 Generation of word graphs

A word graph is a weighted directed acyclic graph, in which each node represents a partial translation hypothesis and each edge is labelled with a word of the target sentence and is weighted according to the scores given by an SMT model (see (Ueffing et al., 2002) for more details). In (Och et al., 2003), the use of a word graph is proposed as interface between an alignment-template SMT model and the IMT engine. Analogously, in this work we will be using a word graph built during the search procedure performed on a PB SMT model.

During the search process performed by the above mentioned beam search algorithm, it is possible to create a *segment graph*. In such a graph, each node represents a state of the SMT model, and each edge a weighted transition between states labelled with a sequence of target words. Whenever a hypothesis is extended, we add a new edge connecting the state of that hypothesis with the state of the extended hypothesis. The new edge is labelled with the sequence of target words that has been incorporated to the extended hypothesis and is weighted appropriately by means of the score given by the SMT model.

Once the segment graph is generated, it can be easily converted into a word graph by the introduction of artificial states for the words that compose the target phrases associated to the edges.

3.2 IMT using word graphs

During the process of IMT for a given source sentence, the system makes use of the word graph generated for that sentence in order to complete the prefixes accepted by the human translator. Specifically,

SOURCE $(\mathbf{x})$**:** Para encender la impresora:
REFERENCE $(\mathbf{y})$**:** To power on the printer:

ITER-0	$(\mathbf{p})$	( )
	$(\hat{\mathbf{s}}_h)$	*To switch on:*
ITER-1	$(\mathbf{p})$	To
	$(\mathbf{s}_l)$	|*switch on:*
	$(\hat{\mathbf{s}}_h)$	*power on the printer:*
ITER-2	$(\mathbf{p})$	To power on the printer:
	$(\mathbf{s}_l)$	( )
	(k)	(#)
	$(\hat{\mathbf{s}}_h)$	( )
FINAL	$(\mathbf{p} \equiv \mathbf{y})$	To power on the printer:

Figure 2: Example of non-explicit positioning MA which solves an error of a missing word. In this case, the system produces the correct suffix $\mathbf{s}_h$ immediately after the user validates a prefix $\mathbf{p}$, implicitly indicating that we wants the suffix to be changed, without need of any further action. In **ITER-1**, character | indicates the position where a MA was performed, $\mathbf{s}_l$ is the suffix which was rejected by that MA, and $\hat{\mathbf{s}}_h$ is the new suffix that the system suggests after observing that $\mathbf{s}_l$ is to be considered incorrect. Character # is a special character introduced by the user to indicate that the hypothesis is to be accepted.

the system finds the best path in the word graph associated with a given prefix so that it is able to complete the target sentence, being capable of providing several completion suggestions for each prefix.

A common problem in IMT arises when the user sets a prefix which cannot be found in the word graph, since in such a situation the system is unable to find a path through the word graph and provide an appropriate suffix. The common procedure to face this problem is to perform a tolerant search in the word graph. This tolerant search uses the well known concept of Levenshtein distance in order to obtain the most similar string for the given prefix (see (Och et al., 2003) for more details).

4 Enriching user–machine interaction

Although the IMT paradigm has proved to offer interesting benefits to potential users, one aspect that has not been reconsidered as of yet is the user–machine interface. Hence, in traditional IMT the system only received feedback whenever the user typed in a new word. In this work, we show how to enrich user–machine interaction by introducing Mouse Actions (MA) as an additional information source for the system. By doing so, we will consider two types of MAs, i.e. *non-explicit* (or *positioning*) MAs and *interaction-explicit* MAs.

4.1 Non-explicit positioning MAs

Before typing in a new word in order to correct a hypothesis, the user needs to position the cursor in the place where he wants to type such a word. In this work, we will assume that this is done by performing a MA, although the same idea presented can also be applied when this is done by some other means. It is important to point out that, by doing so, the user is already providing some very useful information to the system: he is validating a prefix up to the position where he positioned the cursor, and, in addition, he is signalling that whatever word is located after the cursor is to be considered incorrect. Hence, the system can already capture this fact and provide a new translation hypothesis, in which the prefix remains unchanged and the suffix is replaced by a new one in which the first word is different to the first word of the previous suffix. We are aware that this does not mean that the new suffix will be correct, but given that we know that the first word in the previous suffix was incorrect, the worst thing which can happen is that the the first word of the new suffix is incorrect as well. However, if the new suffix happens to be correct, the user will happily find that he does not need to correct that word any more.

An example of such behaviour can be seen in Figure 2. In this example, the SMT system first provides a translation which the user does not like. Hence, he positions the cursor before word "*postscript*", with the purpose of typing in "*lists*". By doing so, he is validating the prefix "*To print a*", and signalling that he wants "*postscript*" to be replaced. Before typing in anything, the system realises that he is going to change the word located after the cursor, and replaces the suffix by another one, which is the one the user had in mind in the first place. Finally, the user only has to accept the final translation.

We are naming this kind of MA *non-explicit* because it does not require any additional action from the user: he has already performed a MA in order to position the cursor at the place he wants, and we are taking advantage of this fact to suggest a new suffix hypothesis.

Since the user needs to position the cursor before typing in a new word, it is important to point out that any improvement achieved by introducing non-explicit MAs does not require any further effort from the user, and hence is considered to have no cost.

Hence, we are now considering two different situations: the first one, the traditional IMT framework, in which the system needs to find a suffix according to Eq. 5, and a new one, in which the system needs to find a suffix in which the first word does not need to be a given k, but needs to be *different* to a given s_{l1}. This constraint can be expressed by the following equation:

$$\hat{\mathbf{s}}_h = \underset{\mathbf{s}_h : s_{h_1} \neq s_{l_1}}{\operatorname{argmax}} \; Pr(\mathbf{p}, \mathbf{s}_h | \mathbf{x}, \mathbf{s}_l) \tag{6}$$

where $\mathbf{s}_l$ is the suffix generated in the previous iteration, already discarded by the user, and s_{l_1} is the first word in $\mathbf{s}_l$. k is omitted in this formula because the user did not type any word at all.

4.2 Interaction-explicit MAs

If the system is efficient and provides suggestions which are good enough, one could easily picture a situation in which the expert would ask the system to replace a given suffix, without typing in any word. We will be modelling this as another kind of MA, *interaction-explicit* MA, since the user needs to indicate *explicitly* that he wants a given suffix to be replaced, in contrast to the non-explicit positioning MA. However, if the underlying MT engine providing the suffixes is powerful enough, the user would quickly realise that performing a MA is less costly that introducing a whole new word, and would take advantage of this fact by systematically clicking before introducing any new word. In this case, as well, we assume that the user clicks before an incorrect word, hence demanding a new suffix whose first word is different, but by doing so he is adopting a more participative and interactive attitude, which was not demanded in the case of non-explicit positioning MAs. An example of such an explicit MA correcting an error can be seen in Figure 3

In this case, however, there is a cost associated to this kind of MAs, since the user does need to perform additional actions, which may or may not be beneficial. It is very possible that, even after asking for several new hypothesis, the user will even though need to introduce the word he had in mind, hence wasting the additional MAs he had performed.

If we allow the user to perform n MAs before introducing a word, this problem can be formalised in an analogous way as in the case of non-explicit MAs as follows:

$$\hat{\mathbf{s}}_h = \underset{\mathbf{s}_h : s_{h_1} \neq s^i_{l_1} \forall i \in \{1..n\}}{\operatorname{argmax}} \; Pr(\mathbf{p}, \mathbf{s}_h | \mathbf{x}, \mathbf{s}^1_l, \mathbf{s}^2_l, \ldots, \mathbf{s}^n_l) \tag{7}$$

where s^i_{l1} is the first word of the i-th suffix discarded and $\mathbf{s}^1_l, \mathbf{s}^2_l, \ldots, \mathbf{s}^n_l$ is the set of all n suffixes discarded.

Note that this kind of MA could also be implemented with some other kind of interface, e.g. by typing some special key such as `F1` or `Tab`. However, the experimental results would not differ, and in our user interface we found it more intuitive to implement it as a MA.

5 Experimental setup

5.1 System evaluation

Automatic evaluation of results is a difficult problem in MT. In fact, it has evolved to a research field with own identity. This is due to the fact that, given an input sentence, a large amount of correct *and* different output sentences may exist. Hence, there is no sentence which can be considered ground truth, as is the case in speech or text recognition. By extension, this problem is also applicable to IMT.

In this paper, we will be reporting our results as measured by *Word Stroke Ratio* (WSR) (Barrachina

SOURCE $(\mathbf{x})$**:** Seleccione el tipo de instalación.
REFERENCE $(\mathbf{y})$**:** Select the type of installation.

ITER-0	$(\mathbf{p})$	( )
	$(\hat{\mathbf{s}}_h)$	*Select the installation wizard.*
ITER-1	$(\mathbf{p})$	Select the
	$(\mathbf{s}_l)$	|*installation wizard.*
	$(\hat{\mathbf{s}}_h)$	*install script.*
ITER-2	$(\mathbf{p})$	Select the
	(k)	type
	$(\hat{\mathbf{s}}_h)$	*installation wizard.*
ITER-3	$(\mathbf{p})$	Select the type
	$(\mathbf{s}_l)$	|*installation wizard.*
	$(\hat{\mathbf{s}}_h)$	*of installation.*
ITER-4	$(\mathbf{p})$	Select the type of installation.
	$(\mathbf{s}_l)$	( )
	(k)	(#)
	$(\hat{\mathbf{s}}_h)$	( )
FINAL	$(\mathbf{p} \equiv \mathbf{y})$	Select the type of installation.

Figure 3: Example of explicit interactive MA which corrects an erroneous suffix. In this case, a non-explicit MA is performed in **ITER-1** with no success. Hence, the user introduces word "*type*" in **ITER-2**, which leaves the cursor position located immediately after word "*type*". In this situation the user would not need to perform a MA to re-position the cursor and continue typing in order to further correct the remaining errors. However, since he has learnt the potential benefit of MAs, he performs an interaction-explicit MA in order to ask for a new suffix hypothesis, which happens to correct the error.

and others, 2008), which is computed as the quotient between the number of word-strokes a user would need to perform in order to achieve the translation he has in mind and the total number of words in the sentence. In this context, a word-stroke is interpreted as a single action, in which the user types a complete word, and is assumed to have constant cost. Moreover, each word-stroke also takes into account the cost incurred by the user when reading the new suffix provided by the system.

In the present work, we decided to use WSR instead of *Key Stroke Ratio* (KSR), which is used in other works on IMT such as (Och et al., 2003). The reason for this is that KSR is clearly an optimistic measure, since in such a scenario the user is often overwhelmed by receiving a great amount of translation options, as much as one per key stroke, and it is not taken into account the time the user would need to read all those hypotheses.

In addition, and because we are also introducing MAs as a new action, we will also present results in terms of *Mouse Action Ratio* (MAR), which is the quotient between the amount of explicit MAs performed and the number of words of the final translation. Hence, the purpose is to elicit the number of times the user needed to request a new translation (i.e. performed a MA), on a per word basis.

Lastly, we will also present results in terms of uMAR (useful MAR), which indicates the amount of MAs which were *useful*, i.e. the MAs that actually produced a change in the first word of the suffix and such word was accepted. Formally, uMAR is defined as follows:

$$uMAR = \frac{MAC - n \cdot WSC}{MAC} \tag{8}$$

where MAC stands for "Mouse Action Count", WSC for "Word Stroke Count" and n is the maximum amount of MAs allowed before the user types in a word. Note that $MAC - n \cdot WSC$ is the amount of MAs that were useful since WSC is the amount of word-strokes the user performed even though he had already performed n MAs.

Since we will only use single-reference WSR and MAR, the results presented here are clearly pessimistic. In fact, it is relatively common to have the underlying SMT system provide a perfectly correct

Table 1: Characteristics of Europarl for each of the sub-corpora. OoV stands for "Out of Vocabulary" words, Dev. for Development, K for thousands of elements and M for millions of elements.

		De	En	Es	En	Fr	En
Training	Sentences	751K		731K		688K	
	Run. words	15.3M	16.1M	15.7M	15.2M	15.6M	13.8M
	Avg. len.	20.3	21.4	21.5	20.8	22.7	20.1
	Voc.	195K	66K	103K	64K	80K	62K
Dev.	Sentences	2000		2000		2000	
	Run. words	55K	59K	61K	59K	67K	59K
	Avg. len.	27.6	29.3	30.3	29.3	33.6	29.3
	OoV	432	125	208	127	144	138
Test	Sentences	2000		2000		2000	
	Run. words	54K	58K	60K	58K	66K	58K
	Avg. len.	27.1	29.0	30.2	29.0	33.1	29.3
	OoV	377	127	207	125	139	133

translation, which is "corrected" by the IMT procedure into another equivalent translation, increasing WSR and MAR significantly by doing so.

5.2 Corpora

Our experiments were carried out on the Europarl (Koehn, 2005) corpus, which is a corpus widely used in SMT and that has been used in several MT evaluation campaigns. Moreover, we performed our experiments on the partition established for the Workshop on Statistical Machine Translation of the NAACL 2006 (Koehn and Monz, 2006). The Europarl corpus (Koehn, 2005) is built from the proceedings of the European Parliament. Here, we will focus on the German–English, Spanish–English and French–English tasks, since these were the language pairs selected for the cited workshop. The corpus is divided into three separate sets: one for training, one for development, and one for test. The characteristics of the corpus can be seen in Table 1.

5.3 Experimental results

As a first step, we built a SMT system for each of the language pairs cited in the previous subsection. This was done by means of the Moses toolkit (Koehn and others, 2007), which is a complete system for building Phrase-Based SMT models. This toolkit involves the estimation from the training set of four different translation models, which are in turn combined in a log-linear fashion by adjusting a weight for each of them by means of the MERT (Och, 2003) procedure, optimising the BLEU (Papineni et al., 2002) score obtained on the development partition.

Table 2: WSR improvement when considering non-explicit MAs. "rel." indicates the relative improvement. All results are given in %.

pair	baseline	non-explicit	rel.
Es–En	63.0±0.9	59.2±0.9	6.0±1.4
En–Es	63.8±0.9	60.5±1.0	5.2±1.6
De–En	71.6±0.8	69.0±0.9	3.6±1.3
En–De	75.9±0.8	73.5±0.9	3.2±1.2
Fr–En	62.9±0.9	59.2±1.0	5.9±1.6
En–Fr	63.4±0.9	60.0±0.9	5.4±1.4

This being done, word graphs were generated for the IMT system. For this purpose, we used a multi-stack phrase-based decoder which will be distributed in the near future together with the Thot toolkit (Ortiz-Martínez et al., 2005). We discarded the use of the Moses decoder because preliminary experiments performed with it revealed that the decoder by (Ortiz-Martínez et al., 2005) performs clearly better when used to generate word graphs for use in IMT. In addition, we performed an experimental comparison in regular SMT with the Europarl corpus, and found that the performance difference was negligible. The decoder was set to only consider monotonic translation, since in real IMT scenarios considering non-monotonic translation leads to excessive waiting time for the user.

Finally, the word graphs obtained were used within the IMT procedure to produce the reference translation contained in the test set, measuring WSR and MAR. The results of such a setup can be seen in Table 2. As a baseline system, we report the traditional IMT framework, in which no MA is taken into account. Then, we introduced non-explicit MAs, obtaining an average improvement in WSR of about 3.2% (4.9% relative). The table also shows the confidence intervals at a confidence level of 95%. These intervals were computed following the bootstrap technique described in (Koehn, 2004). Since the confidence intervals do not overlap, it can be stated that the improvements obtained are statistically significant.

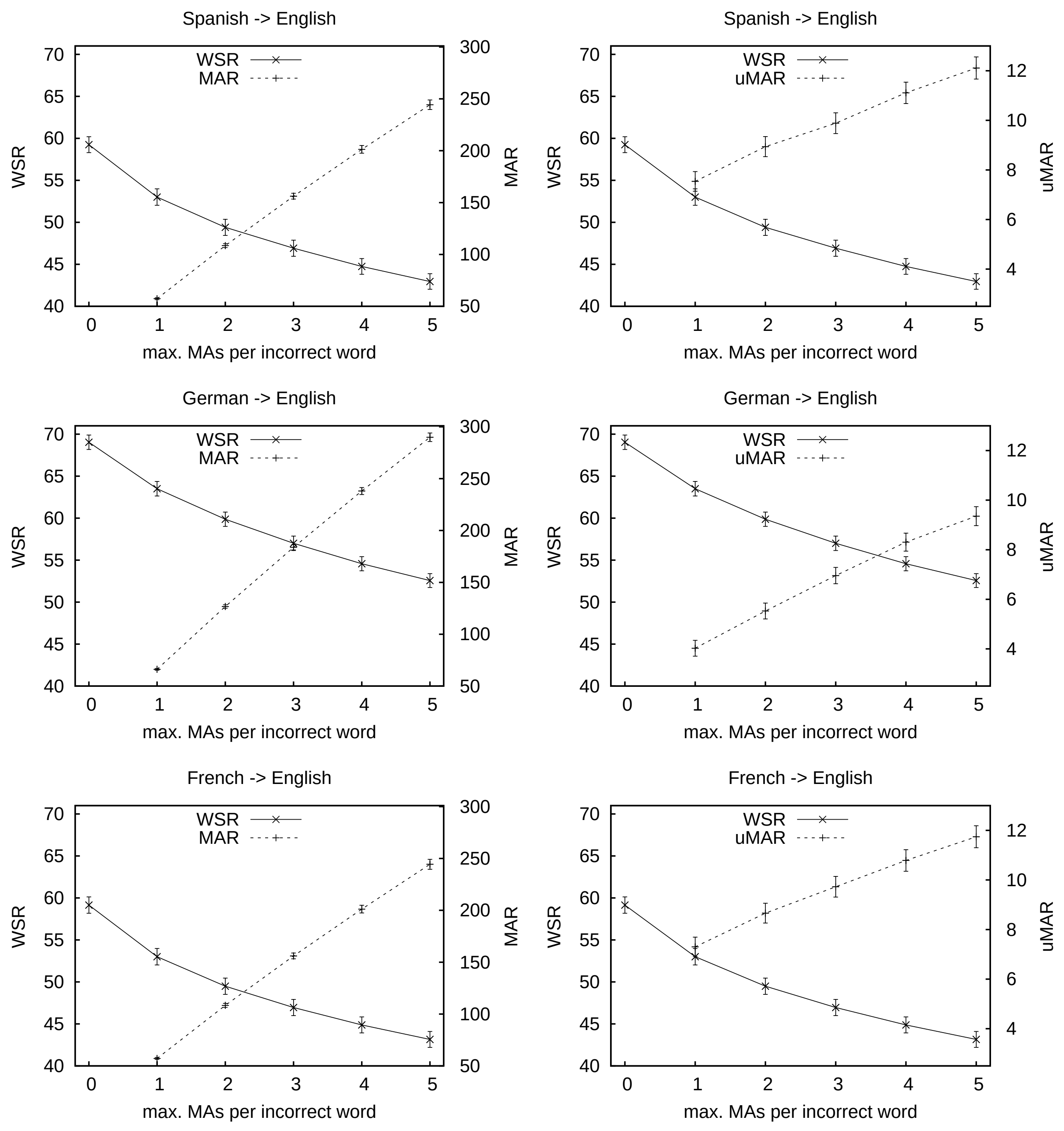


Figure 4: WSR improvement when considering one to five maximum MAs. All figures are given in %. The left column lists WSR improvement versus MAR degradation, and the right column lists WSR improvement versus uMAR. Confidence intervals at 95% confidence level following (Koehn, 2004).

Once the non-explicit MAs were considered and introduced into the system, we analysed the effect of performing up to a maximum of 5 explicit MAs. Here, we modelled the user in such a way that, in case a given word is considered incorrect, he will always ask for another translation hypothesis until he has asked for as many different suffixes as MAs considered. The results of this setup can be seen in Figure 4. This yielded a further average improvement in WSR of about 16% (25% relative improvement) when considering a maximum of 5 explicit MAs. However, relative improvement in WSR and uMAR increase drop significantly when increasing the maximum allowed amount of explicit MAs from 1 to 5. For this reason, it is difficult to imagine that a user would perform more than two or three MAs before actually typing in a new word. Nevertheless, just by asking twice for a new suffix before typing in the word he has in mind, the user might be saving about 15% of word-strokes.

Although the results in Figure 4 are only for the translation direction "foreign"→English, the experiments in the opposite direction (i.e. English→"foreign") were also performed. However, the results were very similar to the ones displayed here. Because of this, and for clarity purposes, we decided to omit them and only display the direction "foreign"→English.

6 Conclusions and future work

In this paper, we have considered new input sources for IMT. By considering Mouse Actions, we have shown that a significant benefit can be obtained, in terms of word-stroke reduction, both when considering only non-explicit MAs and when considering MAs as a way of offering the user several suffix hypotheses. In addition, we have applied these ideas on a state-of-the-art SMT baseline, such as phrase-based models. To achieve this, we have first obtained a word graph for each sentence which is to be translated. Experiments were carried out on a reference corpus in SMT.

Note that there are other systems (Esteban and others, 2004) that, for a given prefix, provide n-best lists of suffixes. However, the functionality of our system is slightly (but fundamentally) different, since the suggestions are demanded to be different in their first word, which implies that the n-best list is scanned deeper, going directly to those hypotheses that may be of interest to the user. In addition, this can be done "on demand", which implies that the system's response is faster and that the user is not confronted with a large list of hypotheses, which often results overwhelming.

As future work, we are planning on performing a human evaluation that assesses the appropriateness of the improvements described.

Acknowledgements

This work has been partially supported by the Spanish MEC under scholarship AP2005-4023 and under grants CONSOLIDER Ingenio-2010 CSD2007-00018, and by the EC (FEDER) and the Spanish MEC under grant TIN2006-15694-CO2-01.

References

D. J. Arnold, 2003. *Computers and Translation: A translator's guide*, chapter 8, pages 119–142.

S. Barrachina et al. 2008. Statistical approaches to computer-assisted translation. *Computational Linguistics*, page In press.

F. Casacuberta and E. Vidal. 2007. Learning finite-state models for machine translation. *Machine Learning*, 66(1):69–91.

J. Esteban et al. 2004. Transtype2 - an innovative computer-assisted translation system. In *The Companion Volume to the Proc. ACL'04*, pages 94–97.

G. Foster, P. Langlais, and G. Lapalme. 2002. User-friendly text prediction for translators. In *Proc. of EMNLP'02*, pages 148–155.

G. Foster. 2002. *Text Prediction for Translators*. Ph.D. thesis, Université de Montréal.

J. Hutchings and H. Somers. 1992. An introduction to machine translation. In *Ed. Academic Press.*

J. Hutchins. 1999. Retrospect and prospect in computer-based translation. In *Proc. of MT Summit VII*, pages 30–44.

M. Kay. 1997. It's still the proper place. *Machine Translation*, 12(1-2):35–38.

P. Koehn and C. Monz, editors. 2006. *Proc. of the Workshop on SMT*.

P. Koehn et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of the ACL'07*.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proc. HLT/NAACL'03*, pages 48–54.

P. Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. of EMNLP'04*, pages 388–395, Barcelona, Spain.

P. Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proc. of the MT Summit X*, pages 79–86.

P. Langlais, G. Lapalme, and M. Loranger. 2002. Transtype: Development-evaluation cycles to boost translator's productivity. *Machine Translation*, 15(4):77–98.

Bruce T. Lowerre. 1976. *The harpy speech recognition system*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, USA.

F. Och and H. Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. of the ACL'02*, pages 295–302.

F.J. Och and H. Ney. 2004. The alignment template approach to statistical machine translation. *Comput. Linguist.*, 30(4):417–449.

F. Och, C. Tillmann, and H. Ney. 1999. Improved alignment models for statistical machine translation. In *Proc. of EMNLP/WVLC'99*, pages 20–28.

F.J. Och, R. Zens, and H. Ney. 2003. Efficient search for interactive statistical machine translation. In *Proc. of EACL'03*, pages 387–393.

F.J. Och. 2003. Minimum error rate training for statistical machine translation. In *Proc. of ACL'03*, pages 160–167.

D. Ortiz-Martínez, I. García-Varea, and F. Casacuberta. 2005. Thot: a toolkit to train phrase-based statistical translation models. In *Proc. of the MT Summit X*, pages 141–148.

K. Papineni, S. Roukos, and T. Ward. 1998. Maximum likelihood and discriminative training of direct translation models. In *Proc. of ICASSP'98*, pages 189–192.

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

N. Ueffing, F. Och, and H. Ney. 2002. Generation of word graphs in statistical machine translation. In *Proc. of EMNLP'02*, pages 156–163.

E. Vidal et al. 2007. Interactive pattern recognition. In *Proc. of MLMI'07*, pages 60–71.

R. Zens, F.J. Och, and H. Ney. 2002. Phrase-based statistical machine translation. In *Proc. of KI'02*, pages 18–32.

LTAG Dependency Parsing with Bidirectional Incremental Construction

Libin Shen
BBN Technologies
lshen@bbn.com

Aravind K. Joshi
University of Pennsylvania
joshi@cis.upenn.edu

Abstract

In this paper, we first introduce a new architecture for parsing, bidirectional incremental parsing. We propose a novel algorithm for incremental construction, which can be applied to many structure learning problems in NLP. We apply this algorithm to LTAG dependency parsing, and achieve significant improvement on accuracy over the previous best result on the same data set.

1 Introduction

The phrase "Bidirectional Incremental" may appear self-contradictory at first sight, since incremental parsing usually means left-to-right parsing in the context of conventional parsing. In this paper, we will extend the meaning of incremental parsing.

The idea of bidirectional parsing is related to the bidirectional sequential classification method described in (Shen et al., 2007). In that paper, a tagger assigns labels to words of highest confidence first, and then these labels in turn serve as the context of later labelling operations. The bidirectional tagger obtained the best results in literature on POS tagging on the standard PTB dataset.

We extend this method from labelling to structure learning, The search space of structure learning is much larger, so that it is appropriate to exploit confidence scores in search.

In this paper, we are interested in LTAG dependency parsing because TAG parsing is a well known problem of high computational complexity in regular parsing. In order to get a focus for the learning algorithm, we work on a variant of LTAG based parsing in which we learn the word dependency relations encoded in LTAG derivations instead of the full-fledged trees.

1.1 Parsing

Two types of parsing strategies are popular in natural language parsing, which are chart parsing and incremental parsing.

Suppose the input sentence is $w_1 w_2 ... w_n$. Let cell $[i, j]$ represent $w_i w_{i+1} ... w_j$, a substring of the sentence. As far as CFG parsing is concerned, a chart parser computes the possible structures over all possible cells $[i, j]$, where $1 \leq i \leq j \leq n$. The order of computing on these $n(n+1)/2$ cells is based on some partial order $\preceq$, such that $[p_1, p_2] \preceq [q_1, q_2]$ if $q_1 \leq p_1 \leq p_2 \leq q_2$. In order to employ dynamic programming, one can only use a fragment of a hypothesis to represent the whole hypothesis, which is assumed to satisfy conditional independence assumption. It is well known that richer context representation gives rise to better parsing performance (Johnson, 1998). However, the need for tractability does not allow much internal information to be used to represent a hypothesis. The designs of hypotheses in (Collins, 1999; Charniak, 2000) show a delicate balance between expressiveness and tractability, which play an important role in natural language parsing.

Some recent work on incremental parsing (Collins and Roark, 2004; Shen and Joshi, 2005) showed another way to handle this problem. In these incremental parsers, tree structures are used to represent the left context. In this way, one can access the whole tree to collect rich context information at the expense of being limited to beam search, which only maintains k-best results at each step. Compared to chart parsing, incremental parsing searches for the analyses for only $2n-1$ cells, $[1,1],[2,2],[1,2],..,[i,i],[1,i],..,[1,n]$, incrementally, while complex structures are used for the analyses for each cell, which satisfy conditional independence under a much weaker assumption.

In this paper, we call this particular approach **left-to-right** incremental parsing, since one can also search from right to left incrementally in a similar way. A major problem of the left-to-right approach is that one can only utilize the structural information on the left side but not the right side.

1.2 Parsing as Bidirectional Construction

A natural way to handle this problem is to employ bidirectional search, which means we can dynamically search the space in two directions. So we expand the idea of incremental parsing by introducing greedy search. Specifically, we look for the hypotheses over the cell $[1,n]$ by building analyses over $2n-1$ cells $[a_{i,1},a_{i,2}], i=1,..,2n-1$ step by step, where $[a_{2n-1,1},a_{2n-1,2}]=[1,n]$. Furthermore, for any $[a_{i,1},a_{i,2}]$

- $a_{i,1}=a_{i,2}$, or
- $\exists j,k,$ such that $[a_{i,1},a_{i,2}]=[a_{j,1},a_{k,2}]$, where $j<i$, $k<i$ and $a_{j,2}+1=a_{k,1}$.

It is easy to show that the set $\{[a_{i,1},a_{i,2}] \mid 1 \leq i \leq 2n-1\}$ forms a *tree* relation, which means that each cell except the last one will be used to build another cell just once. In this framework, we can begin with several starting points in a sentence and search in any direction. So left-to-right parsing is only a special case of **incremental** parsing defined in this way. We still use complex structures to represent the partial analyses, so as to employ both top-down and bottom-up information as in (Collins and Roark, 2004; Shen and Joshi, 2005). Furthermore, we can utilize the rich context on both sides of the partial results.

Similar to bidirectional labelling in (Shen et al., 2007), there are two learning tasking in this model. First, we need to learn which cell we should choose. At each step, we can select only one path. Secondly, we need to learn which operation we should take for a given cell. We maintain k-best candidates for each cell instead of only one, which differentiates this model from normal greedy search. So our model is more robust. Furthermore, we need to find an effective way to iterate between these two tasks.

Instead of giving an algorithm specially designed for parsing, we generalize the problem for graphs. A sentence can be viewed as a graph in which words are viewed as vertices and neighboring words are connected with an arc. In Sections 2 and 3, we will propose decoding and training algorithms respectively for graph-based incremental construction, which can be applied to many structure learning problems in NLP.

We will apply this algorithm to dependency parsing of Lexicalized Tree Adjoining Grammar (Joshi and Schabes, 1997). Specifically, we will train and evaluate an LTAG dependency parser over the LTAG treebank described in Shen et al. (2008). We report the experimental results on PTB section 23 of the LTAG treebank. The accuracy on LTAG dependency is 90.5%, which is 1.2 points over 89.3%, the previous best result (Shen and Joshi, 2005) on the same data set.

It should be noted that PTB-based bracketed labelling is not an appropriate evaluation metric here, since the experiments are on an LTAG treebank. The derived trees in the LTAG treebank are different from the CFG trees in PTB. Hence, we do not use metrics such as labeled precision and labeled recall for evaluation.

2 Graph-based Incremental Construction

2.1 Idea and Data Structures

Now we define the problem formally. We will use dependency parsing as an example to illustrate the idea.

We are given a connected graph $G(V,E)$ whose hidden structure is U, where $V=\{v_i\}$, $E \subseteq V \times V$ is a symmetric relation, and $U=\{u_k\}$ is composed of a set of elements that vary with applications. As far as dependency parsing is concerned, the input graph is simply a chain of vertices, where $E(v_{i-1},v_i)$, and its hidden structure is $\{u_k=(v_{s_k},v_{e_k},b_k)\}$, where vertex v_{e_k} depends on vertex v_{s_k} with label b_k.

A graph-based incremental construction algorithm looks for the hidden structure in a bottom-up style.

Let x_i and x_j be two sets of connected vertexes in V, where $x_i \cap x_j = \phi$ and they are directly connected via an edge in E. Let y^{xi} be a hypothesized hidden structure of x_i, and y^{xj} a hypothesized hidden structure of x_j.

Suppose we choose to combine y^{xi} and y^{xj} with an operation r to build a hypothesized hidden structure for $x_k = x_i \cup x_j$. We say the process of construction is **incremental** if the output of the operation, $y^{xk} = r(x_i, x_j, y^{xi}, y^{xj}) \supseteq y^{xi} \cup y^{xj}$ for all the possible x_i, x_j, y^{xi}, y^{xj} and operation r. As far as dependency parsing is concerned, incrementality means that we cannot remove any links coming from the substructures.

Once y^{xk} is built, we can no longer use y^{xi} or y^{xj} as a building block. It is easy to see that left to right incremental construction is a special case of our approach. So the question is how we decide the order of construction as well as the type of operation r. For example, in the very first step of dependency parsing, we need to decide which two words are to be combined as well as the dependency label to be used.

This problem is solved statistically, based on the features defined on the substructures involved in the operation and their context. Suppose we are given the weights of these features, we will show in the next section how these parameters guide us to build a set of hypothesized hidden structures with beam search. In Section 3, we will present a Perceptron like algorithm (Collins, 2002; Daumé III and Marcu, 2005) to obtain the parameters.

Now we introduce the data structure to be used in our algorithms.

A **fragment** is a *connected* sub-graph of $G(V, E)$. Each fragment x is associated with a set of hypothesized hidden structures, or **fragment hypotheses** for short: $Y^x = \{y_1^x, ..., y_k^x\}$. Each y^x is a possible fragment hypothesis of x.

It is easy to see that an operation to combine two fragments may depend on the fragments in the context, i.e. fragments directly connected to one of the operands. So we introduce the **dependency** relation over fragments. Suppose there is a dependency relation $D \subseteq F \times F$, where $F \subseteq 2^V$ is the set of all fragments in graph G. $D(x_i, x_j)$ means that any operation on a fragment hypothesis of x_i depends on the features in the fragment hypothesis of x_j, and vice versa.

We are especially interested in the following two dependency relations.

- *level-0 dependency*: $D_0(x_i, x_j) \iff i = j$.
- *level-1 dependency*: $D_1(x_i, x_j) \iff x_i$ and x_j are directly connected in G.

Level-0 dependency means that the features of a hypothesis for a vertex x_i do not depend on the hypotheses for other vertices. Level-1 dependency means that the features depend on the hypotheses of nearby vertices only.

The learning algorithm for level-0 dependency is similar to the guided learning algorithm for labelling as described in (Shen et al., 2007). Level-1 dependency requires more data structures to maintain the hypotheses with dependency relations among them. However, we do not get into the details of level-1 formalism in this papers for two reasons. One is the limit of page space and depth of a conference paper. On the other hand, our experiments show that the parsing performance with level-1 dependency is close to what level-0 dependency could provides. Interested readers could refer to (Shen, 2006) for detailed description of the learning algorithms for level-1 dependency.

2.2 Algorithms

Algorithm 1 shows the procedure of building hypotheses incrementally on a given graph $G(V, E)$. Parameter k is used to set the beam width of search. Weight vector $\mathbf{w}$ is used to compute score of an operation.

We have two sets, H and Q, to maintain hypotheses. Hypotheses in H are selected in beam search, and hypotheses in Q are candidate hypotheses for the next step of search in various directions.

We first initiate the hypotheses for each vertex, and put them into set H. For example, in dependency parsing, the initial value is a set of possible POS tags for each single word. Then we use a queue Q to collect all the possible hypotheses over the initial hypotheses H.

Whenever Q is not empty, we search for the hypothesis with the highest score according to a given weight vector $\mathbf{w}$. Suppose we find (x, y). We select

Algorithm 1 Incremental Construction

```
Require: graph $G(V, E)$;
Require: beam width $k$;
Require: weight vector $\mathbf{w}$;
1: $H \leftarrow init_H()$;
2: $Q \leftarrow init_Q(H)$;
3: repeat
4:     $(x', y') \leftarrow arg\max_{(x,y)\in Q} score(y)$;
5:     $H \leftarrow update_H(H, x')$;
6:     $Q \leftarrow update_Q(Q, H, x')$;
7: until $(Q = \phi)$
```

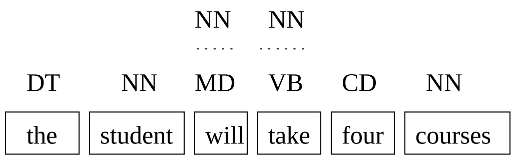


Figure 1: After initialization

top k-best hypotheses for segment x from Q and use them to update H. Then we remove from Q all the hypotheses for segments that have overlap with segment x. In the end, we build new candidate hypotheses with the updated selected hypothesis set H, and add them to Q.

2.3 An Example

We use an example of dependency parsing to illustrate the incremental construction algorithm first.

Suppose the input sentence is *the student will take four courses*. We are also given the candidate POS tags for each word. So the graph is just a linear structure in this case. We use level-0 dependency and set beam width to two.

We use boxes to represent fragments. The dependency links are from the parent to the child.

Figure 1 shows the result after initialization. Figure 2 shows the result after the first step, combining the fragments of *four* and *courses*. Figure 3 shows the result after the second step, combining *the* and *student*, and figure 4 shows the result after the third step, combining *take* and *four courses*. Due to limited space, we skip the rest operations.

2.4 Description

Now we will explain the functions in Algorithm 1 one by one.

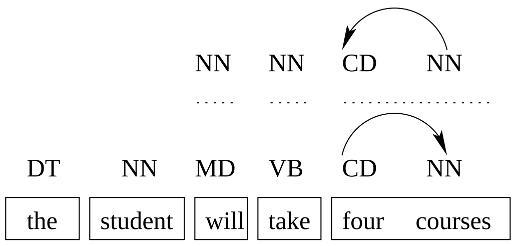


Figure 2: Step 1

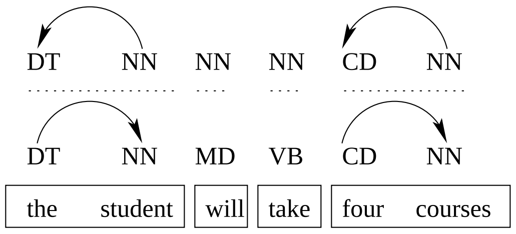


Figure 3: Step 2

- $init_H()$ initiates hypotheses for each vertex. Here we set the initial fragment hypotheses, $Y^{xi} = \{y_1^{xi}, ..., y_k^{xi}\}$, where $x_i = \{v_i\}$ contains only one vertex.

- $init_Q(H)$ initiates the queue of candidate operations over the current hypotheses H. Supposed there exist segments x_i and x_j which are directly connected in G. We apply all possible operations to all fragment hypotheses for x_j and x_j, and add the result hypotheses in Q. For example, we generate (x, y) with some operation r, where segment x is $x_i \cup x_j$.

  All the candidate operations are organized with respect to the segments. For each segment, we maintain top k candidates according to their scores.

- $update_H(H, x)$ is used to update hypotheses in H. First, we remove from H all the hypotheses whose corresponding segment is a sub-set of x. Then, we add into H the top k hypotheses for segment x.

- $update_Q(Q, H, x)$ is also designed to complete two tasks. First, we remove from Q all the hypotheses whose corresponding segment has overlap with segment x. Then, we add new candidate hypotheses depending on x in a way

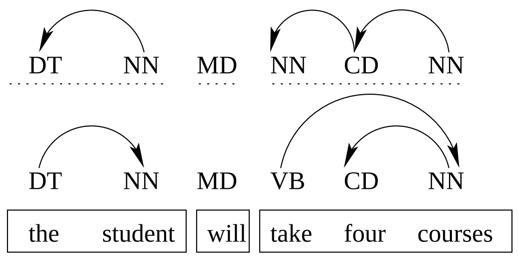


Figure 4: Step 3

Algorithm 2 Parameter Optimization

```
1:  $\mathbf{w} \leftarrow \mathbf{0}$;
2:  for (round $r = 0$; $r < R$; $r$++) do
3:      load graph $G_r(V, E)$, gold standard $H_r$;
4:      initiate $H$ and $Q$;
5:      repeat
6:          $(x', y') \leftarrow arg\max_{(x,y)\in Q} score(y)$;
7:          if ($y'$ is compatible with $H_r$) then
8:              update $H$ and $Q$;
9:          else
10:             $\tilde{y} \leftarrow positive(Q, x')$;
11:             $promote(\mathbf{w}, \tilde{y})$;
12:             $demote(\mathbf{w}, y')$;
13:             update $Q$ with $\mathbf{w}$;
14:         end if
15:     until ($Q = \phi$)
16: end for
```

similar to the $init_Q(H)$ function. For each segment, we maintain the top k candidates for each segment.

3 Parameter Optimization

In the previous section, we described an algorithm for graph-based incremental construction for a given weight vector **w**. In Algorithm 2, we present a Perceptron like algorithm to obtain the weight vector for the training data.

For each given training sample (G_r, H_r), where H_r is the gold standard hidden structure of graph G_r, we first initiate cut T, hypotheses H^T and candidate queue Q by calling $init_H$ and $init_Q$ as in Algorithm 1.

Then we use the gold standard H_r to guide the search. We select candidate (x', y') which has the highest operation score in Q. If y' is compatible with H_r, we update H and Q by calling $update_H$ and $update_Q$ as in Algorithm 1. If y' is incompatible with H_r, we treat y' as a negative sample, and search for a positive sample $\tilde{y}$ in Q with $positive(Q, x')$.

If there exists a hypothesis $\tilde{y}^{x'}$ for fragment x' which is compatible with H_r, then $positive(Q, x')$ returns $\tilde{y}^{x'}$. Otherwise $positive(Q, x')$ returns the candidate hypothesis which is compatible with H_r and has the highest operation score in Q.

Then we update the weight vector $\mathbf{w}$ with $\tilde{y}$ and y'. At the end, we update the candidate Q by using the new weights $\mathbf{w}$.

In order to improve the performance, we use *Perceptron with margin* in the training (Krauth and Mézard, 1987). The margin is proportional to the loss of the hypothesis. Furthermore, we use averaged weights (Collins, 2002; Freund and Schapire, 1999) in Algorithm 1.

4 LTAG Dependency Parsing

We apply the new algorithm to LTAG dependency parsing on an LTAG Treebank (Shen et al., 2008) extracted from Penn Treebank (Marcus et al., 1994) and Proposition Bank (Palmer et al., 2005). Penn Treebank was previously used to train and evaluate various dependency parsers (Yamada and Matsumoto, 2003; McDonald et al., 2005). In these works, Magerman's rules are used to pick the head at each level according to the syntactic labels in a local context.

The dependency relation encoded in the LTAG Treebank reveals deeper information for the following two reasons. First, the LTAG architecture itself reveals deeper dependency. Furthermore, the PTB was reconciled with the Propbank in the LTAG Treebank extraction (Shen et al., 2008).

We are especially interested in the two types of structures in the LTAG Treebank, predicate adjunction and predicate coordination. They are used to encode dependency relations which are unavailable in other approaches. On the other hand, these structures turn out to be a big problem for the general representation of dependency relations, including adjunction and coordination. We will show that the algorithm proposed here provides a nice solution for this problem.

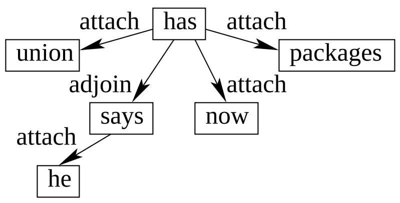


Figure 5: Predicate Adjunction

4.1 Representation of the LTAG Treebank

In the LTAG Treebank (Shen et al., 2008), each word is associated with a spinal template, which represents the projection from the lexical item to the root. Templates are linked together to form a *derivation* tree. The topology of the derivation tree shows a type of dependency relation, which we call **LTAG dependency** here.

There are three types of operations in the LTAG Treebank, which are attachment, adjunction, and coordination. Attachment is used to represent both substitution and sister adjunction in the traditional LTAG. So it is similar to the dependency relation in other approaches.

The LTAG dependency can be a non-projective relation thanks to the operation of adjunction. In the LTAG Treebank, raising verbs and passive ECM verbs are represented as auxiliary trees to be adjoined. In addition, adjunction is used to handle many cases of discontinuous arguments in Propbank. For example, in the following sentence, ARG1 of *says* in Propbank is discontinuous, which is *First Union now has packages for seven customer groups*.

- *First Union, he says, now has packages for seven customer groups.*

In the LTAG Treebank, the subtree for *he says* adjoins onto the node of *has*, which is the root of the derivation tree, as shown in Figure 5.

Another special aspect of the LTAG Treebank is the representation of predicate coordination. Figure 6 is the representation of the following sentence.

- *I couldn't resist rearing up on my soggy loafers and saluting.*

The coordination between *rearing* and *saluting* is represented explicitly with a coord-structure, and

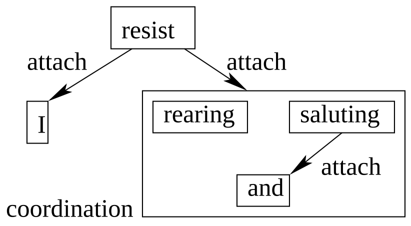


Figure 6: Predicate Coordination

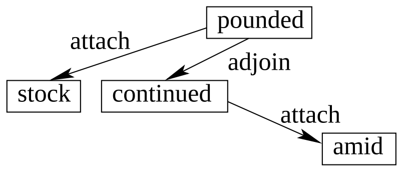


Figure 7: Non-projective Adjunction

this coord-structure attaches to *resist*. It is shown in (Shen et al., 2008) that coord-structures could encode the ambiguity of argument sharing, which can be non-projective also.

4.2 Incremental Construction

We build LTAG derivation trees incrementally. A hypothesis of a fragment is represented with a partial derivation tree. When the fragment hypotheses of two nearby fragments combine, the partial derivation trees are combined into one.

It is trivial to combine two partial derivation trees with attachment. We simply attach the root of one tree to some node on the other tree which is *visible* to this root node. Adjunction is similar to attachment, except that an adjoined subtree may be *visible* from the other side of the derivation tree. For example, in sentence

- *The stock of UAL Corp. continued to be pounded amid signs that British Airways ...*

continued adjoins onto *pounded*, and *amid* attaches to *continued* from the other side of the derivation tree (*pounded* is between *continued* and *amid*), as shown in Figure 7.

The predicate coordination is decomposed into a set of operations to meet the need for incremental processing. Suppose a coordinated structure attaches to the parent node on the left side. We build this structure incrementally by attaching the first

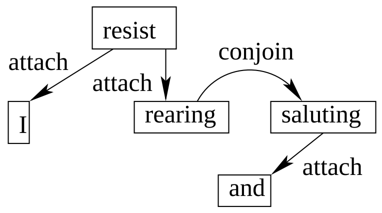


Figure 8: Conjunction

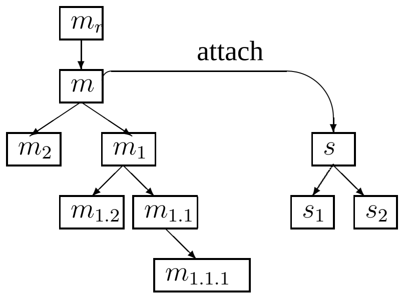


Figure 9: Representation of nodes

conjunct to the parent and conjoining other conjuncts to first one. In this way, we do not need to force the coordination to be built before the attachment. Either can be executed first. A sample is shown in Figure 8.

4.3 Features

In this section, we will describe the features used in LTAG dependency parsing. An operation is represented by a 4-tuple

- $op = (type, dir, pos_{left}, pos_{right})$,

where $type \in \{attach, adjoin, conjoin\}$ and dir is used to represent the direction of the operation. pos_{left} and pos_{right} are the POS tags of the two operands.

Features are defined on POS tags and lexical items of the nodes in the context. In order to represent the features, we use m for the main-node of the operation, s for the sub-node, m_r for the parent of the main-node, $m_1..m_i$ for the children of m, and $s_1..s_j$ for the children of s, as shown in Figure 9. The index always starts from the side where the operation takes place. We use the Gorn addresses to represent the nodes in the subtrees rooted on m and s.

Furthermore, we use l_k and r_k to represent the nodes in the left and right context of the flat sentence. We use h_l and h_r to represent the head of the hypothesis trees on the left and right context respectively. Let x be a node. We use $x.p$ to represent the POS tag of node x, and $x.w$ to represent the lexical item of node x.

Table 1 show the features used in LTAG dependency parsing. There are seven classes of features. The first three classes of features are those defined on only one operand, on both operands, and on the siblings respectively. If gold standard POS tags are used as input, we define features on the POS tags in the context. If level-1 dependency is used, we define features on the root node of the hypothesis partial derivation trees in the neighborhood.

Half check and full check features are designed for grammatical check. For example, in Figure 9, node s attaches onto node m from left. Then nothing can attach onto s from the right side. The children of the right side of s are fixed, so we use the half check features to check the completeness of the children of the right half for s. Furthermore, we notice that all the rightmost descendants of s and the leftmost descendants of m at each level become unavailable for any further operation. So their children are fixed after this operation. All these nodes are in the form of $m_{1.1...1}$ or $s_{1.1...1}$. We use full check features to check the children from both sides for these nodes.

In the discussion above, we ignored adjunction and conjunction. We need to slightly refine the conditions of checking. Due to the limit of space, we skip these cases.

5 Experiments

We use the same data set as in (Shen and Joshi, 2005). We use Sec. 2-21 of the LTAG Treebank for training, Sec. 22 for feature selection, and Sec. 23 for test. Table 2 shows the comparison of different models. Beam size is set to five in our experiments. With level-0 dependency, our system achieves an accuracy of 90.3% at the speed of 4.25 sentences a second on a Xeon 3G Hz processor with JDK 1.5. With level-1 dependency, the parser achieves 90.5% at 3.59 sentences a second. Level-1 dependency does not provide much improvement due to the fact that level-0 features provide most of the useful information for this specific application.

It is interesting to compare our system with other dependency parsers. The accuracy on LTAG depen-

category	description	templates
one operand	Features defined on only one operand. For each template tp, $[type, dir, tp]$ is used as a feature.	$(m.p)$, $(m.w)$, $(m.p, m.w)$, $(s.p)$, $(s.w)$, $(s.p, s.w)$
two operands	Features defined on both operands. For each template tp, $[op, tp]$ is used as a feature. In addition, $[op]$ is also used as a feature.	$(m.w)$, $(s.w)$, $(m.w, s.w)$
siblings	Features defined on the children of the main nodes. For each template tp, $[op, tp]$, $[op, m.w, tp]$, $[op, m_r.p, tp]$ and $[op, m_r.p, m.w, tp]$ are used as features.	$(m_1.p)$, $(m_1.p, m_2.p)$, .., $(m_1.p, m_2.p, .., m_i.p)$
POS context	In the case that gold standard POS tags are used as input, features are defined on the POS tags of the context. For each template tp, $[op, tp]$ is used as a feature.	$(l_2.p)$, $(l_1.p)$, $(r_1.p)$, $(r_2.p)$, $(l_2.p, l_1.p)$, $(l_1.p, r_1.p)$, $(r_1.p, r_2.p)$
tree context	In the case that level-1 dependency is employed, features are defined on the trees in the context. For each template tp, $[op, tp]$ is used as a feature.	$(h_l.p)$, $(h_r.p)$
half check	Suppose $s_1, ..., s_k$ are all the children of s which are between s and m in the flat sentence. For each template tp, $[tp]$ is used as a feature.	$(s.p, s_1.p, s_2.p, .., s_k.p)$, $(m.p, s.p, s_1.p, s_2.p, .., s_k.p)$ and $(s.w, s.p, s_1.p, s_2.p, .., s_k.p)$, $(s.w, m.p, s.p, s_1.p, s_2.p, .., s_k.p)$ if $s.w$ is a verb
full check	Let x_1, x_2, .., x_k be the children of x, and x_r the parent of x. For any $x = m_{1.1...1}$ or $s_{1.1...1}$, template tp, $[tp(x)]$ is used as a feature.	$(x.p, x_1.p, x_2.p, .., x_k.p)$, $(x_r.p, x.p, x_1.p, x_2.p, .., x_k.p)$ and $(x.w, x.p, x_1.p, x_2.p, .., x_k.p)$, $(x.w, x_r.p, x.p, x_1.p, x_2.p, .., x_k.p)$ if $x.w$ is a verb

Table 1: Features defined on the context of operation

model	accuracy%
Shen and Joshi, 2005	89.3
level-0 dependency	90.3
level-1 dependency	90.5

Table 2: Experiments on Sec. 23 of the LTAG Treebank

dency is comparable to the numbers of the previous best systems on dependency extracted from PTB with Magerman's rules, for example, 90.3% in (Yamada and Matsumoto, 2003) and 90.9% in (McDonald et al., 2005). However, their experiments are on the PTB, while ours is on the LTAG corpus.

It should be noted that it is more difficult to learn LTAG dependencies. Theoretically, the LTAG dependencies reveal deeper relations. Adjunction can lead to non-projective dependencies, and the dependencies defined on predicate adjunction are linguistically more motivated, as shown in the examples in Figure 5 and 7. The explicit representation of predicate coordination also provides deeper relations. For example, in Figure 6, the LTAG dependency contains $resist \rightarrow rearing$ and $resist \rightarrow saluting$, while the Magerman's dependency only contains $resist \rightarrow rearing$. The explicit representation of predicate coordination will help to solve for the dependencies for shared arguments.

6 Discussion

In our approach, each fragment in the graph is associated with a hidden structure, which means that we cannot reduce it to a labelling task. Therefore, the problem of interest to us is different from previous work on graphical models, such as CRF (Lafferty et al., 2001) and MMMN (Taskar et al., 2003).

McAllester et al. (2004) introduced Case-Factor Diagram (CFD) to transform a graph based construction problem to a labeling problem. However, adjunction, prediction coordination, and long distance dependencies in LTAG dependency parsing make it difficult to implement. Our approach provides a novel alternative to CFD.

Our learning algorithm stems from Perceptron training in (Collins, 2002). Variants of this method have been successfully used in many NLP tasks, like shallow processing (Daumé III and Marcu, 2005), parsing (Collins and Roark, 2004; Shen and Joshi, 2005) and word alignment (Moore, 2005). Theoretical justification for those algorithms can be applied to our training algorithm in a similar way.

In our algorithm, dependency is defined on complicated hidden structures instead of on a graph. Thus long distance dependency in a graph becomes local in hidden structures, which is desirable from linguistic considerations.

The search strategy of our bidirectional dependency parser is similar to that of the bidirectional CFG parser in (Satta and Stock, 1994; Ageno and Rodrguez, 2001; Kay, 1989). A unique contribution of this paper is that selection of path and decisions about action are trained simultaneously with discriminative learning. In this way, we can employ context information more effectively.

7 Conclusion

In this paper, we introduced bidirectional incremental parsing, a new architecture of parsing. We proposed a novel algorithm for graph-based incremental construction, and applied this algorithm to LTAG dependency parsing, revealing deep relations, which are unavailable in other approaches and difficult to learn. We evaluated the parser on an LTAG Treebank. Experimental results showed significant improvement over the previous best system. Incremental construction can be applied to other structure learning problems of high computational complexity, for example, such as machine translation and semantic parsing.

References

A. Ageno and H. Rodrguez. 2001. Probabilistic modelling of island-driven parsing. In *International Workshop on Parsing Technologies*.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*.

M. Collins and B. Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

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

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference of Empirical Methods in Natural Language Processing*.

H. Daumé III and D. Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of the 22nd International Conference on Machine Learning*.

Y. Freund and R. E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

M. Johnson. 1998. PCFG Models of Linguistic Tree Representations. *Computational Linguistics*, 24(4).

A. K. Joshi and Y. Schabes. 1997. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 69 – 124. Springer-Verlag.

M. Kay. 1989. Head-driven parsing. In *Proceedings of Workshop on Parsing Technologies*.

W. Krauth and M. Mézard. 1987. Learning algorithms with optimal stability in neural networks. *Journal of Physics A*, 20:745–752.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmentation and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

D. McAllester, M. Collins, and F. Pereira. 2004. Case-factor diagrams for structured probabilistic modeling. In *UAI 2004*.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43th Annual Meeting of the Association for Computational Linguistics (ACL)*.

R. Moore. 2005. A discriminative framework for bilingual word alignment. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*.

M. Palmer, D. Gildea, and P. Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1).

G. Satta and O. Stock. 1994. Bi-Directional Context-Free Grammar Parsing for Natural Language Processing. *Artificial Intelligence*, 69(1-2).

L. Shen and A. K. Joshi. 2005. Incremental LTAG Parsing. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*.

L. Shen, G. Satta, and A. K. Joshi. 2007. Guided Learning for Bidirectional Sequence Classification. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*.

L. Shen, L. Champollion, and A. K. Joshi. 2008. LTAG-spinal and the Treebank: a new resource for incremental, dependency and semantic parsing. *Language Resources and Evaluation*, 42(1):1–19.

L. Shen. 2006. *Statistical LTAG Parsing*. Ph.D. thesis, University of Pennsylvania.

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin markov networks. In *Proceedings of the 17th Annual Conference Neural Information Processing Systems*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with Support Vector Machines. In *IWPT 2003*.

Improved Sentence Alignment on Parallel Web Pages Using a Stochastic Tree Alignment Model

Lei Shi
Microsoft Research Asia
5F Sigma Center, 49 Zhichun Road, Beijing
100190, P. R. China
leishi@microsoft.com

Ming Zhou
Microsoft Research Asia
5F Sigma Center, 49 Zhichun Road, Beijing
100190, P. R. China
mingzhou@microsoft.com

Abstract

Parallel web pages are important source of training data for statistical machine translation. In this paper, we present a new approach to sentence alignment on parallel web pages. Parallel web pages tend to have parallel structures，and the structural correspondence can be indicative information for identifying parallel sentences. In our approach, the web page is represented as a tree, and a stochastic tree alignment model is used to exploit the structural correspondence for sentence alignment. Experiments show that this method significantly enhances alignment accuracy and robustness for parallel web pages which are much more diverse and noisy than standard parallel corpora such as “Hansard”. With improved sentence alignment performance, web mining systems are able to acquire parallel sentences of higher quality from the web.

1 Introduction

Sentence-aligned parallel bilingual corpora have been essential resources for statistical machine translation (Brown et al. 1993), and many other multi-lingual natural language processing applications. The task of aligning parallel sentences has received considerable attention since the renaissance of data driven machine translation in late 1980s.

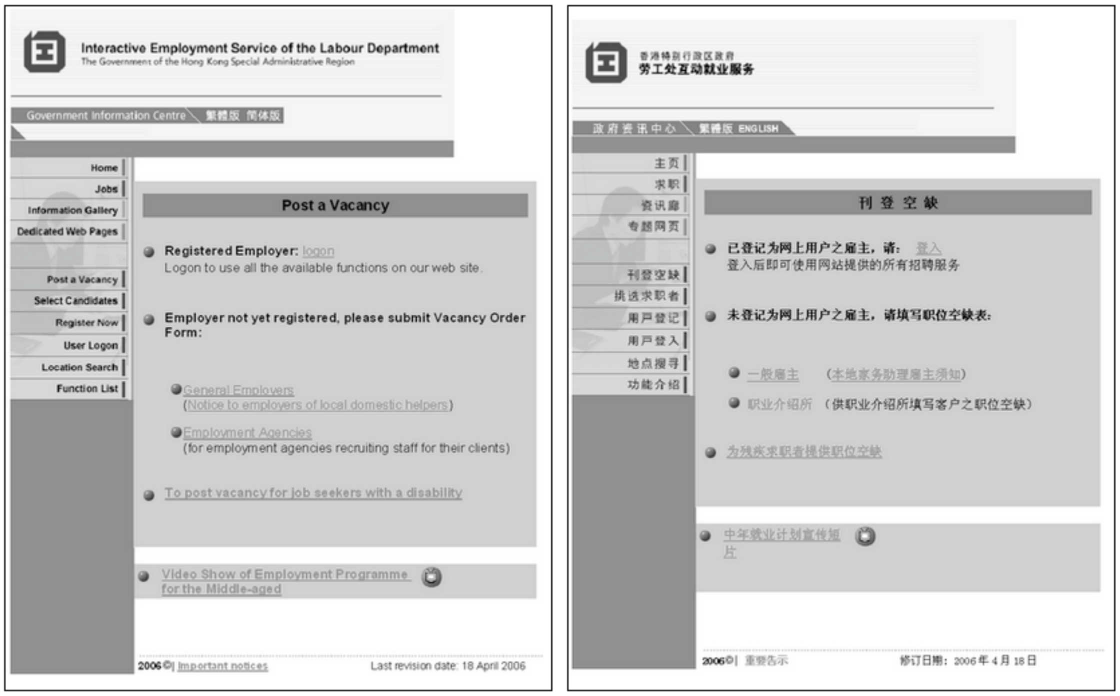
During the past decades, a number of methods have been proposed to address the sentence alignment problem. Although excellent performance was reported on clean corpora, they are less robust with presence of noise. A recent study by (Singh and Husain 2005) completed a systematic evaluation on different sentence aligners under various conditions. Their experiments showed that the performance of sentence aligners are sensitive to properties of the text, such as format complexity (presence of elements other than text), structural distance (a scale from literal to free translation), the amount of noise (text deletions or preprocessing errors) and typological distance between languages. Their performance varies on different type of texts and they all demonstrate marked performance degradation over noisy data. The results suggest that there is currently no universal solution to sentence alignment under all conditions, and different methods should be applied to different types of texts.

In this paper, we specifically address sentence alignment on parallel web pages. It has come to attention with the increasing trend of acquiring large-scale parallel data from the web. Currently, large-scale parallel data are not readily available for most language pairs and domains. But due to a sharply increasing number of bilingual web sites, web mining shows great promise as a solution to this knowledge bottleneck problem. Many systems (Ma 1999; Chen 2000; Yang 2002; Resnik 2003; Chen 2004) have been developed to discover parallel web pages, and sentence aligners are used to extract parallel sentences from the mined web corpora. Sentence alignment performance on parallel web pages, therefore, becomes an increasingly important issue for large-scale high-quality parallel data acquisition.

Compared with clean parallel corpora such as "Hansard" (Brown et al. 1993), which consists of

French-English translations of political debates in the Canadian parliament, texts from the web are far more diverse and noisy. They are from many different domains and of various genres. Their translation may be non-literal or written in disparate language pairs. Noise is abundant with frequent insertions, deletions or non-translations. And there are many very short sentences of 1-3 words. Due to the characteristics of web corpora, direct application of conventional alignment methods without exploiting additional web document information acts as useful information to constrain the scope of search for parallel sentences.

The paper is organized as follows: In section 2, we briefly survey previous approaches to sentence alignment. In section 3, we present the stochastic tree alignment model, including parameter estimation and decoding. Then in section 4, we describe how to use the tree alignment model in sentence alignment. Benchmarks are shown in section 5, and the paper is concluded in section 6.



Registered Employer: logon
Logon to use all the available functions on our web site.

Employer not yet registered, please submit Vacancy Order Form:

General Employers
(Notice to employers of local domestic helpers)

Employment Agencies
(for employment agencies recruiting staff for their clients)

To post vacancy for job seekers with a disability

Video Show of Employment Programme for the Middle-aged

2006©| Important notices Last revision date: 18 April 2006

已登记为网上用户之雇主，请： 登入
登入后即可使用网站提供的所有招聘服务

未登记为网上用户之雇主，请填写职位空缺表：

一般雇主 （本地家务助理雇主须知）

职业介绍所 （供职业介绍所填写客户之职位空缺）

为残疾求职者提供职位空缺

中年就业计划宣传短片

2006©| 重要告示 修订日期：2006年4月18日

Figure 1. Example of parallel web pages

yields unsatisfactory alignment results.

Our approach to this problem is to make use of the structural parallelism between parallel web pages. Structural parallelism is the phenomenon, that when representing the same content in two different languages, authors have a very strong tendency to use the same document structure. As is shown in Figure 1, sentences located in similar position on both pages are more likely to be translations. Hence, correspondence in the web page structure is an informative indication of parallel sentences. In our approach, the web page is represented as a tree, and a stochastic tree alignment model is used to find the most probable alignment of the tree pair based on their structure and the texts in tree nodes. The tree alignment then

2 Sentence Alignment Models

Sentence alignment methods can be categorized into three major categories: the length-based, lexicon-based and hybrid method which combines the length-based model and lexicon-based model as complement to each other.

The length model was based on the intuition that the length of a translated sentence is likely to be similar to that of the source sentence. (Brown et. at. 1991) used word count as the sentence length, whereas (Gale and Church 1993) used character count. Dynamic programming is used to search the optimal sentence alignment. Both algorithms have achieved remarkably good results for language pairs like English-French and English-German with an error rate of 4% on average. But they are not robust with respect to non-literal translations, deletions and disparate language pairs.

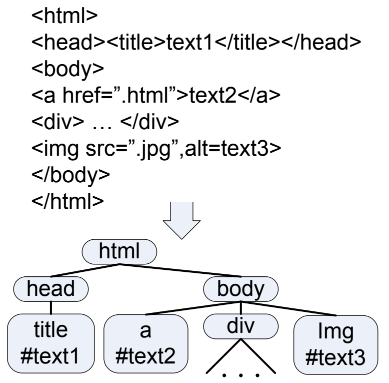
Unlike the length-based model, which totally ignores word identity, lexicon-based methods use lexical information to align parallel sentences. Kay's (Kay and Roscheisen 1993) approach is based on the idea that words that are translations of each other will have similar distribution in source and target texts. By adopting the IBM model 1, (Chen 1993) used word translation probabilities, which he showed gives better accuracy than the sentence length based method. Melamed (Melamed 1996) rather used word correspondence from a different perspective as geometric correspondence for sentence alignment.

The hybrid method combines the length model with the lexical method. (Simard and Plamondon 1996) used a two-pass approach, where the first pass performs length-based alignment at the character level as in (Gale and Church 1993) and the second pass uses IBM Model 1, following (Chen 1993). Moore's (Moore 2002) approach is similar to Simard's. The difference is that Moore used the data obtained in the first pass to train the IBM model in the second pass, so that his approach does not require a priori knowledge about the language pair. Instead of using a two-pass approach, (Zhao and Vogel 2002) combines the length model and the IBM model 1 in a unified framework under a maximum likelihood criterion. To make it more robust on noisy text, they developed a background model to handle text deletions.

To further improve sentence alignment accuracy and robustness, methods that make use of additional language or corpus specific information were developed. In Brown and Church's length-based aligner, they assume prior alignment on some corpus specific anchor points to constrain and keep the Viterbi search on track. (Wu 1994) implemented a length-based model for Chinese-English with language specific lexical clues to improve accuracy. (Simard et al. 1992) used cognates, which only exists in closely related language pairs. (Chuang and Yeh 2005) exploited the statistically ordered matching of punctuation marks in two languages to achieve high accuracy sentence alignment. In their web parallel data mining system, (Chen and Nie 2000) used HTML tags in the same way as cognates in (Simard et al. 1992) for aligning Chinese-English parallel sentences. Tree based alignment models have been successfully applied in machine translation (Wu 1997, Yamada & Knight 2001, Gildea 2003).

3 The Stochastic Tree Alignment Model

The structure of the HTML document is recursive, with HTML markup tags embedded within other markup tags. While converting an HTML document into the tree representation, such hierarchical order is maintained. Each node of the tree is labeled with their corresponding HTML tag (*e.g. body, title, img etc.*) and in labeling tree nodes, only markup tags are used and attribute value pairs are dropped. Among all markup tags in the HTML file, those of our most interest are tags containing content text, which is what we want to align. These tags are those surrounding a text chunk or have the attribute of "ALT". Comments, scripts and style specifications are not regarded as content text and hence are eliminated. Figure 2 illustrates the tree representation of an example HTML document.



```
<html>
<head><title>text1</title></head>
<body>
<a href=".html">text2</a>
<div> … </div>
<img src=".jpg",alt=text3>
</body>
</html>
```

Figure. 2 An example HTML document and its tree representation

3.1 Tree Alignment Model

Given two trees, the tree alignment is the non-directional alignments of their nodes. A node in one tree can be aligned with at most one node in the other tree. It is valid for a node to be aligned with nothing (*NULL*) and such case is regarded as node deletion in tree alignment. To comply with the tree hierarchical structure, we constrain that the alignments keep the tree hierarchy invariant *i.e.* if node *A* is aligned with node *B*, then the children of *A* are either deleted or aligned with the children of *B*. Besides, to simplify the model training and decoding, the tree alignment model also keeps the sequential order invariant, *i.e.* if node A is aligned with node B, then the left sibling nodes of A cannot be aligned with the right sibling nodes of B.

The stochastic tree alignment model assigns probabilities to tree alignments, based on the particular configuration of the alignment and model parameters. Then, the decoder is able to find the most probable (optimal) alignment of two trees. To facilitate the presentation of the tree alignment model, the following symbols are introduced: given a HTML document D, T^D denotes the corresponding tree; N_i^D denotes the ith node of T^D, and T_i^D denotes the sub-tree rooted at N_i^D. Especially, T_1^D is the root of the tree T^D. $T_{[i,j]}^D$ denotes the forest consisting of the sub-trees rooted at sibling nodes from T_i^D to T_j^D. $N_i^D.t$ denotes the text in the node N_i^D, and $N_i^D.t$ denotes the label (*i.e.* HTML tag) of the node N_i^D; $N_i^D.C_i$ denotes the jth child of the node N_i^D; $N_i^D.C_{[m,n]}$ denotes the consecutive sequence of N_i^D's children nodes from $N_i^D.C_m$ to $N_i^D.C_n$; the sub-tree rooted at $N_i^D.C_i$ is represented as $N_i^D.CT_i$ and the forest of the sub-trees rooted at N_i^D's children is represented as $N_i^D.CF$. To accommodate node deletion, $NULL$ is introduced to denote the empty node. Finally, the tree alignment is referred as A.

Given two HTML documents F (in French) and E (in English) represented as trees T^F and T^E, the tree alignment task is defined as finding the alignment A that maximizes the conditional probability $\Pr(A|T^F,T^E)$. Based on the Bayes' Rule, $\Pr(A|T^F,T^E) \propto \Pr(T^F,T^E|A)\Pr(A)$, where $\Pr(T^F,T^E|A)$ is the probability of synchronously generating T^F and T^E given the alignment A, and $\Pr(A)$ is the prior knowledge of the tree alignment. To simplify computation, we assume a uniform prior probability $\Pr(A)$. Hence, the tree alignment task is to find the A that maximizes the synchronous probability $\Pr(T^F,T^E|A)$.

Based on the hierarchical structure of the tree, in order to facilitate the presentation and computation of the tree alignment probabilistic model, the following alignment probabilities are defined in a hierarchically recursive manner:

$\Pr(T_m^F,T_i^E|A)$: The probability of synchronously generating sub-tree pair $\{T_m^F,T_i^E\}$ given the alignment A;

$\Pr(N_m^F,N_i^E|A)$: The probability of synchronously generating node pair $\{N_m^F,N_i^E\}$;

$\Pr(T_{[m,n]}^F,T_{[i,j]}^E|A)$: The probability of synchronously generating forest pairs $\{T_{[m,n]}^F,T_{[i,j]}^E\}$ given the alignment A.

From the definition, the tree pair generative probability $\Pr(T^F,T^E|A)$ equals to the root sub-tree pair generative probability $\Pr(T_1^F,T_1^E|A)$. The alignment of the sub-tree pair T_j^F and T_i^E may have the following configurations, based on which the tree pair generative probability $\Pr(T_j^F,T_i^E|A)$ can be calculated:

(1) If N_m^F is aligned with N_i^E, and the children of N_m^F are aligned with children of N_i^E (as is shown in Fig. 3a), then we have

$$\Pr(T_m^F,T_i^E|A) = \Pr(N_m^F,N_i^E)\Pr(N_m^F.CF,N_i^E.CF|A)$$

(2) If N_m^F is deleted, and the children of N_m^F is aligned with N_i^E (as shown in Fig. 3b), then we have

$$\Pr(T_m^F,T_i^E|A) = \Pr(N_m^F|NULL)\Pr(N_m^F.CF,T_i^E|A)$$

(3) If N_i^E is deleted, and N_m^F is aligned with children of N_i^E (as shown in Fig. 3c), then we have

$$\Pr(T_m^F,T_i^E|A) = \Pr(T_m^F,N_i^E.CF|A)\Pr(N_i^E|NULL)$$

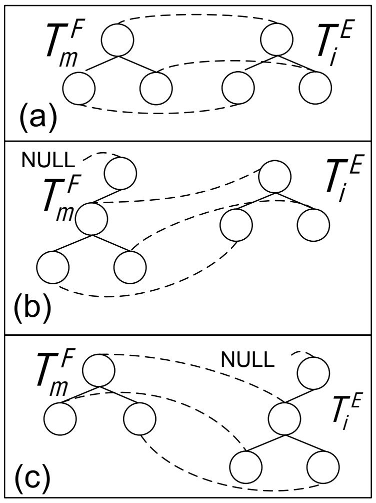


Figure. 3

The above equations involve forest pair generative probabilities. The alignment of the forest $T_{[m,n]}^F$ and $T_{[i,j]}^E$ may have the following configurations, based on which their forest pair generative probability $\Pr(T_{[m,n]}^F,T_{[i,j]}^E|A)$ can be calculated:

(4) If T_m^F is aligned with T_i^E, and $T_{[m+1,n]}^F$ is aligned with $T_{[i+1,j]}^E$ (as is shown in Fig. 4a), then

$$\Pr(T_{[m,n]}^F, T_{[i,j]}^E | A) = \Pr(T_m^F, T_i^E | A)\Pr(T_{[m+1,n]}^F, T_{[i+1,j]}^E | A)$$

(5) If N_m^F is deleted, and the forest rooted at N_m^F's children $N_m^F.CF$ is combined with $T_{[m+1,n]}^F$ for alignment with $T_{[i,j]}^E$, then

$$\Pr(T_{[m,n]}^F, T_{[i,j]}^E | A) = \Pr(N_m^F | NULL)\Pr(N_m^F.CF\ T_{[m+1,n]}^F, T_{[i,j]}^E | A)$$

(6) If N_i^E is deleted, and the forest rooted at N_i^E's children $N_i^E.CF$ is combined with $T_{[i,j]}^E$ for alignment with $T_{[m,n]}^F$, then

$$\Pr(T_{[m,n]}^F, T_{[i,j]}^E | A) = \Pr(N_i^E | NULL)\Pr(T_{[m,n]}^F, N_m^F.CF\ T_{[i+1,j]}^E | A)$$

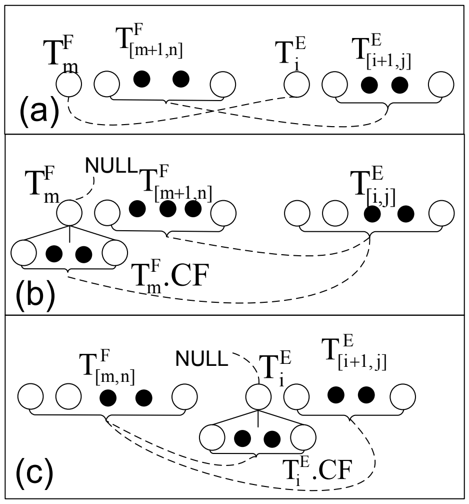


Figure. 4

Finally, the node pair probability is modeled as $\Pr(N_m^F, N_i^E) = \Pr(N_m^F.t, N_i^E.t)\Pr(N_m^F.l, N_i^E.l)$, where $\Pr(N_m^F.t, N_i^E.t)$ is the generative probability of the translationally equivalent text chunks in N_m^F and N_i^E, and $\Pr(N_m^F.l, N_i^E.l)$ is their HTML tag pair probability. The text chunk generative probability $\Pr(N_m^F.t, N_i^E.t)$ can be modeled in a variety of ways. The conventional length-based, lexicon-based or hybrid methods used for sentence alignment can be applied here. In the next sub-section, we focus on how to estimate the tag pair probability $\Pr(N_m^F.l, N_i^E.l)$ from a set of parallel web pages. We expect pairs of the same or similar HTML tags to have high probabilities and the probabilities for pairs of disparate tags to be low.

3.2 Parameter Estimation Using Expectation-Maximization

One way to estimate the tag pair generative probability $\Pr(l, l')$ is to manually align nodes between parallel trees, and use the manually aligned trees as the training data for maximum likelihood estimation. However, this is a time-consuming and error-prone procedure. Instead, the Expectation Maximization (EM) (Dempster, Laird and Rubin 1977) algorithm is used to estimate the parameters $\Pr(l, l')$ on 5615 manually verified parallel web page pairs from 45 different bilingual web sites. The parameter estimation proceeds as follows:

1. Start with initial parameter values.
2. Expectation: estimate $\overline{\text{count}}(l, l')$ which is the expectation of aligning tag l with l'.
3. Maximization: update the parameters based to maximum likelihood estimation
$$\Pr(l, l') = \frac{\overline{count}(l, l')}{\sum_{l'} \overline{count}(l, l')}$$ and
$$\Pr(l | NULL) = \frac{\overline{count}(NULL, l) + \overline{count}(l, NULL)}{\sum_{l'} [\overline{count}(NULL, l') + \overline{count}(l', NULL)]}$$
4. Repeat step 2 and 3 until the parameters stabilize

In step 2, $\overline{\text{count}}(l, l')$ is the expected count of l being aligned with l' in the training corpus. By definition, $\overline{\text{count}}(l, l')$ is calculated as

$$\overline{\text{count}}(l, l') = \sum_A \Pr(A | T^F, T^E)\text{count}(l, l')$$

where $\text{count}(l, l')$ is the number of occurrence of l being aligned with l' in the tree alignment A.

To efficiently compute $\overline{\text{count}}(l, l')$ without enumerating the exponential number of A's in the above equation, we extended the inside-outside algorithm presented in (Lari and Young, 1990). The inside probability $\alpha(N_j^F, N_i^E)$ is defined as the probability of generating sub-tree pair $\{T_j^F, T_i^E\}$ when N_i^E is aligned with N_j^F. It is estimated as:

$$\alpha\left(N_m^F, N_i^E\right) = \Pr\left(N_m^F, N_i^E\right)\alpha\left(N_m^F.CF, N_i^E.CF\right)$$

where $\alpha(N_m^F.CF, N_i^E.CF)$ is the inside probability for the forest pair $(N_m^F.CF, N_i^E.CF)$

$$\alpha\left(N_m^F.CF, N_i^E.CF\right) = \sum_A \Pr\left(N_m^F.CF, N_i^E.CF \middle| A\right).$$

The inside probability can be estimated recursively according to the various alignment configurations presented in Figure 3 and Figure 4. The outside probability $\beta(N_j^F, N_i^E)$ is defined as the probability of generating the part of T^E and T^F excluding the sub-trees T_j^F and T_i^E, when N_i^E is aligned with N_j^F. It is estimated as:

$$\begin{aligned}\beta\left(N_m^F, N_i^E\right) = \sum_{p,q} [\beta\left(a_{m,q}^F, a_{i,p}^E\right) \\ \times \alpha\left(a_{m,q}^F.LCF\left(N_m^F\right), a_{i,p}^E.LCF\left(N_i^E\right)\right) \\ \times \alpha\left(a_{m,q}^F.RCF\left(N_m^F\right), a_{i,p}^E.RCF\left(N_i^E\right)\right) \\ \times \prod_{k<q} \Pr(a_{m,k}^F \mid NULL) \prod_{k<p} \Pr(a_{i,k}^E \mid NULL)]\end{aligned}$$

where $a_{m,q}^F$ is the qth ancestor of N_m^F, and $a_{i,q}^E$ is the pth ancestor of N_i^E. $a_{m,k}^F (k < q)$ is an ancestor of N_m^F and a decedent of $a_{m,q}^F$. Similarly $a_{i,k}^E (k <$ p is an ancestor of *NiE*, and a decedent of *ai,pE*. *a.LCF(N)* is the forest rooted at *a* and to the left of *N*, and *a.RCF(N)*. *a.RCF(N)* is the forest rooted as *a* and to the right of *N*. Once inside and outside probabilities are computed, the expected counts can be calculated as

$$\overline{count}\left(l, l'\right) = \sum_{\{T^F, T^E\}} \sum_{\substack{N_i^E.l=l \\ N_m^F.l=l}} \frac{\alpha\left(N_m^F, N_i^E\right)\beta\left(N_m^F, N_i^E\right)}{\Pr(T^F, T^E)}$$

where $\Pr(T^F, T^E)$ is the generative probability of the tree pair $\{T^F, T^E\}$ over all possible alignment configurations. $\Pr(T^F, T^E)$ can be estimated using dynamic programming techniques that will be presented in the next sub-section. Furthermore, the expected count of tag deletion is estimated as:

$$\overline{\mathrm{count}}(l, NULL) = \sum_i \mathrm{count}(l, l') - \sum_{i \neq NULL} \overline{\mathrm{count}}(l, l')$$

$$\overline{\mathrm{count}}(NULL, l) = \sum_i \mathrm{count}(l', l) - \sum_{i \neq NULL} \overline{\mathrm{count}}(l', l)$$

3.3 Dynamic Programming for Decoding

An intuitive way to find the optimal tree alignment is to enumerate all alignments and pick the one with the highest probability. But it is intractable since the total number of alignments is exponential. Based on the observation that if two trees are optimally aligned, the alignment of their sub-trees must also be optimal, dynamic programming can be applied to find the optimal tree alignment using that of the sub-trees in a bottom-up manner. That is we first compute the optimal alignment probabilities of small trees and use them to compute that of the bigger tree by trying different alignment configurations. This procedure is recursive until the optimal alignment probability of the whole tree is obtained. The following is the pseudo-code of the bottom-up decoding algorithm:

```
for i=$|T^E|$ to 1 (bottom-up) {
  for j=$|T^F|$ to 1 (bottom-up) {
    Select and store optimal alignments of their children forests $T_m^F$.CF and $T_i^E$.CF by testing configurations 4-6;
    Select and store the optimal alignment of the sub-tree pair $T_m^F$ and $T_i^E$ by testing configurations 1-3;
    Store the optimal configuration}}
```

where $|T^F|$ and $|T^E|$ are the number of nodes in T^F and T^E. The decoding algorithm finds the optimal alignment and its probability for every sub-trees and forests. By replacing the selection operation with summing probabilities of all configurations, the sub-tree pair generative probability $\Pr(T^F, T^E)$ can be calculated along the way. The worst-case time complexity of the algorithm is $O(|T^F||T^E|\left(degr(T^F) + degr(T^E)\right)^2)$, where the degree of a tree is defined as the largest degree of its nodes.

4 Sentence Alignment with Tree Alignment Model

Since the tree alignment model aligns parallel web pages at the tree node level instead of the sentence level, we integrate the tree alignment model with the sentence alignment model in a cascaded mode, in which the whole sentence alignment process is divided into two steps. In the first step, the tree alignment decoder finds the optimal alignment of the two trees. Nodes having texts should be aligned with nodes containing their translations. Then in the second step, the conventional sentence aligner is used to align sentences within text chunks in the aligned nodes. In this step, various sentence alignment models can be applied, including the length-based model, the lexicon-based model and the hybrid model. Language or corpus specific information may also be used to further improve sentence alignment accuracy. The tree alignment acts as constraints that confine the scope of the search of sentence aligners.

5 Evaluation

To evaluate the effectiveness of exploiting web page document structure with the tree alignment model for improving sentence alignment accuracy, we compared the performance of three types of sentence alignment methods on parallel web pages.

The first type is to simply discard web page layout information. Web pages are converted to plain texts, and HTML tags are removed prior to performing sentence alignment. The second type is the baseline method of using web page document information. Instead of exploiting full HTML document structure, it follows Chen's approach (Chen and Nie 2000) which uses HTML tags in the same way as cognates used in (Simard et al. 1992). The third type is the combination of tree alignment model and conventional sentence models.

computer and literature. By manual annotation, 9,824 parallel sentence pairs are found. All sentence aligners run through the test parallel web pages, and each extracts a set of sentence pairs that it regards as parallel. The output pairs are matched with the annotated parallel sentences from the test corpus. Only exact matches of the sentence pairs are counted as correct.

Our evaluation metrics are precision (P), recall (R) and F-measure (F) defined as:

$$\mathrm{P} = \frac{\#\text{ of correctly aligned sentence pairs}}{\#\text{ of total output pairs}}$$

$$\mathrm{R} = \frac{\#\text{ of correctly aligned sentence pairs}}{\#\text{ of true parallel pairs}}$$

$$\mathrm{F} = \frac{2 * \mathrm{P} * \mathrm{R}}{\mathrm{P} + \mathrm{R}}$$

Based on the results in table 1, we can see that both Type 2 and Type 3 aligners outperform conventional sentence alignment models. Leveraging HTML document information can enhance sentence alignment quality. Especially, by using the tree alignment model, Type 3 aligners achieve a

	Length			Lexicon			Hybrid		
	P	R	F	P	R	F	P	R	F
Type I	85.6%	72.8%	78.7%	83.1%	75.2%	78.9%	87.3%	76.4%	81.5%
Type II	86.3%	74.8%	80.1%	85.7%	77.0%	81.1%	88.1%	78.6%	83.1%
Type III	93.2%	79.3%	85.7%	92.9%	80.4%	86.2%	94.3%	83.1%	88.3%

Table 1. Performance comparison between different types of sentence alignment methods

Each type of the web page sentence aligner makes use of three conventional sentence alignment models, one is the length based model following (Brown 1991), one is the lexicon based model following (Chen 1993), and the other one is the hybrid model presented in (Zhao 2002). To be fair in performance comparisons, the text generative probability $\Pr(N^F.t, N^E.t)$ in tree node alignment is modeled in accordance with that in the sentence alignment model. All these sentence aligners are implemented to handle sentence bead types of "1-0", "0-1","1-1", "1-2","1-3","2-1" and "3-1".

The test corpus is 150 parallel web page pairs randomly drawn from 20 Chinese-English bilingual web sites on topics related to politics, sports,

significant increase of around 7% on both precision and recall. Compared with the tree alignment model, the improvement by the Type 2 aligners is marginal. A reason for this is that the tree alignment model not only exploits HTML tag similarities as in the Type 2 method, but also takes into account location of texts. In the tree alignment model, texts at similar locations in the tree hierarchical structure are more probable to be translations than those in disparate locations, even though they all have the same tag.

We also evaluate the performance of the tree aligner. Since sentence alignment is performed within the text chunks of aligned nodes, tree alignment accuracy is very important for correct sentence alignment. We measure the alignment accuracy on all nodes as well as that specifically on text nodes on the test corpus. The evaluation result is shown in table 2.

	total	correct	accuracy
all node alignment	18492	17966	97.2%
text node alignment	3646	3577	98.1%

Table 2. Tree Alignment Metrics

Benchmarks in Table 2 show that the tree alignment model yields very reliable results with high accuracy in aligning both text nodes and non-text nodes. After an analysis on text node alignment errors, we find that 79.7% of them have texts of very short length (no more than 4 words), which may not contain sufficient information to be identified as parallel.

6 Conclusions

In this paper, we present a new approach to sentence alignment on parallel web pages. Due to the diversity and noisy nature of web corpora, a stochastic tree alignment model is employed to exploit document structure in parallel web pages as useful information for identifying parallel sentences. The tree alignment model can be combined with various conventional sentence alignment models to extract parallel sentences from parallel web pages. Experimental results show that exploiting structural parallelism inherent in parallel web pages provides superior alignment performance over conventional sentence alignment methods and significant improvement (around 7% in both precision and recall) is achieved by using the stochastic tree alignment model. With improved sentence alignment performance, web parallel data mining systems are able to acquire parallel sentences of higher quality and quantity from the web.

References:

Brown, P. F., J. C. Lai and R. L. Mercer. 1991. *Aligning Sentences in Parallel Corpora.* Proceedings of ACL 1991.

Brown, P. E., S. A. D. Pietra, V. J. D. Pietra, and R. L. Mercer. 1993. *The Mathematics of Statistical Machine Translation: Parameter Estimation*, Computational Linguistics V19(2), 1993

Chen Jisong., Chau R. and C.-H. Yeh. 2004. *Discovering Parallel Text from the World Wide Web*, Proceedings of the second workshop on Australasian Information Security, Data Mining and Web Intelligence, and Software Internationalization.

Chen Jiang and Nie Jianyun. 2000. *Automatic construction of parallel English-Chinese corpus for cross-language information retrieval.* Proceedings of the sixth conference on applied natural language processing

Chen Stanley. 1993. *Aligning Sentences in Bilingual Corpora Using Lexical Information.* Proceedings of ACL 1993

Chuang T.C. and Yeh.K.C. 2005. *Aligning Parallel Bilingual Corpora Statistically with Punctuation Criteria.* Computational Linguistics and Chinese Language Processing. Vol. 10, 2005, pp. 95-122

Dempster, A., Laird, N., and Rubin, D. 1977. *Maximum likelihood from incomplete data via the EM algorithm.* Journal of the Royal Statistical Society, Series B, 39(1):1–38.

Gale W. A. and K. Church. 1993. *A Program for Aligning Sentences in Parallel Corpora*, Computational Linguistics, 19(1):75—102

Gildea. D. 2003. *Loosely Tree-Based Alignment for Machine Translation.* In Proceedings of ACL 2003

Kay Martin and Roscheisen Martin 1993. *Text Translation Alignment.* Computational Linguistics 19(1):121--142.

Lari K. and S. J. Young. 1990. *The estimation of stochastic context free grammars using the Inside-Outside algorithm,* Computer Speech and Language, 4:35—56

Ma, Xiaoyi and M. Liberman. 1999. *Bits: A Method for Bilingual Text Search over the Web.* Proceedings of Machine Translation Summit VII.

Melamed. I. Dan. 1996. *A Geometric Approach to Mapping Bitext Correspondence.* Proceedings of EMNLP 96

Moore Robert. C. 2002. *Fast and Accurate Sentence Alignment of Bilingual Corpora.* Proceedings of 5th Conference of the Association for Machine Translation in the Americas, pp. 135-244

Resnik, P. and N.A. Smith. 2003. *The Web as a Parallel Corpus.*Computational Linguistics, 29(3)

Simard, M. and Plamondon, P. 1996 *Bilingual Sentence Alignment: Balancing Robustness and Accuracy.* Proceedings of AMTA-96, Canada.

Simard, M., Foster, G. and Isabelle, P. 1992, *Using Cognates to Align Sentences in Bilingual Corpora.* Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine translation (TMI92)

Singh, A. K. and Husain, S. (2005). *Comparison, selection and use of sentence alignment algorithms for new language pairs.* Proceedings of the ACL Workshop on Building and Using Parallel Texts.

Wu. Dekai. 1994. *Aligning a parallel English-Chinese corpus statistically with lexical criterias.* Proceedings of ACL 1994.

Wu. Dekai. "*Stochastic Inversion Transduction Grammar and Bilingual Parsing of Parallel Corpora*" Computational Linguistics, 23(3):374(1997)

Yamada H. and Knight K. 2001 *A Syntax based statistical translation model*. In Proceedings of ACL-01

Yang C. C., and Li K. W., *Mining English/Chinese Parallel Documents from the World Wide Web,* Proceedings of the International World Wide Web Conference, Honolulu, Hawaii, 2002.

Zhao Bin. and Stephan. Vogel. 2002. *Adaptive Parallel Sentences Mining From Web Bilingual News Collection.* 2002 IEEE International Conference on Data Mining. 745-748

HTM: A Topic Model for Hypertexts

Congkai Sun*
Department of Computer Science
Shanghai Jiaotong University
Shanghai, P. R. China
martinsck@hotmail.com

Bin Gao
Microsoft Research Asia
No.49 Zhichun Road
Beijing, P. R. China
bingao@microsoft.com

Zhenfu Cao
Department of Computer Science
Shanghai Jiaotong University
Shanghai, P. R. China
zfcao@cs.sjtu.edu.cn

Hang Li
Microsoft Research Asia
No.49 Zhichun Road
Beijing, P. R. China
hangli@microsoft.com

Abstract

Previously topic models such as PLSI (Probabilistic Latent Semantic Indexing) and LDA (Latent Dirichlet Allocation) were developed for modeling the contents of *plain texts*. Recently, topic models for processing *hypertexts* such as web pages were also proposed. The proposed hypertext models are generative models giving rise to both *words* and *hyperlinks*. This paper points out that to better represent the contents of hypertexts it is more essential to assume that the hyperlinks are fixed and to define the topic model as that of generating words only. The paper then proposes a new topic model for hypertext processing, referred to as Hypertext Topic Model (HTM). HTM defines the distribution of words in a document (i.e., the content of the document) as a mixture over latent topics in the document *itself* and latent topics in the documents which *the document cites*. The topics are further characterized as distributions of words, as in the conventional topic models. This paper further proposes a method for learning the HTM model. Experimental results show that HTM outperforms the baselines on topic discovery and document classification in three datasets.

1 Introduction

Topic models are probabilistic and generative models representing contents of documents. Examples of topic models include PLSI (Hofmann, 1999) and LDA (Blei et al., 2003). The key idea in topic modeling is to represent topics as distributions of words and define the distribution of words in document (i.e., the content of document) as a mixture over hidden topics. Topic modeling technologies have been applied to natural language processing, text mining, and information retrieval, and their effectiveness have been verified.

In this paper, we study the problem of topic modeling for *hypertexts*. There is no doubt that this is an important research issue, given the fact that more and more documents are available as hypertexts currently (such as web pages). Traditional work mainly focused on development of topic models for plain texts. It is only recently several topic models for processing hypertexts were proposed, including Link-LDA and Link-PLSA-LDA (Cohn and Hofmann, 2001; Erosheva et al., 2004; Nallapati and Cohen, 2008).

We point out that existing models for hypertexts may not be suitable for *characterizing contents of hypertext documents*. This is because all the models are assumed to generate both words and hyperlinks (outlinks) of documents. The generation of the latter type of data, however, may not be necessary for the tasks related to contents of documents.

In this paper, we propose a new topic model for hypertexts called HTM (Hypertext Topic Model), within the Bayesian learning approach (it is similar to LDA in that sense). In HTM, the hyperlinks of hypertext documents are supposed to be given. Each document is associated with one topic distribution. The word distribution of a document is defined as a mixture of latent topics of the document *itself* and latent topics of documents *which the document cites*. The topics are further defined as distributions

* This work was conducted when the first author visited Microsoft Research Asia as an intern.

of words. That means the content (topic distributions for words) of a hypertext document is not only determined by the topics of itself but also the topics of documents it cites. It is easy to see that HTM contains LDA as a special case. Although the idea of HTM is simple and straightforward, it appears that this is the first work which studies the model.

We further provide methods for learning and inference of HTM. Our experimental results on three web datasets show that HTM outperforms the baseline models of LDA, Link-LDA, and Link-PLSA-LDA, in the tasks of topic discovery and document classification.

The rest of the paper is organized as follows. Section 2 introduces related work. Section 3 describes the proposed HTM model and its learning and inference methods. Experimental results are presented in Section 4. Conclusions are made in the last section.

2 Related Work

There has been much work on topic modeling. Many models have been proposed including PLSI (Hofmann, 1999), LDA (Blei et al., 2003), and their extensions (Griffiths et al., 2005; Blei and Lafferty, 2006; Chemudugunta et al., 2007). Inference and learning methods have been developed, such as variational inference (Jordan et al., 1999; Wainwright and Jordan, 2003), expectation propagation (Minka and Lafferty, 2002), and Gibbs sampling (Griffiths and Steyvers, 2004). Topic models have been utilized in topic discovery (Blei et al., 2003), document retrieval (Xing Wei and Bruce Croft, 2006), document classification (Blei et al., 2003), citation analysis (Dietz et al., 2007), social network analysis (Mei et al., 2008), and so on. Most of the existing models are for processing plain texts. There are also models for processing hypertexts, for example, (Cohn and Hofmann, 2001; Nallapati and Cohen, 2008; Gruber et al., 2008; Dietz et al., 2007), which are most relevant to our work.

Cohn and Hofmann (2001) introduced a topic model for hypertexts within the framework of PLSI. The model, which is a combination of PLSI and PHITS (Cohn and Chang, 2000), gives rise to both the words and hyperlinks (outlinks) of the document in the generative process. The model is useful when the goal is to understand the distribution of links as well as the distribution of words. Erosheva et al (2004) modified the model by replacing PLSI with LDA. We refer to the modified mode as Link-LDA and take it as a baseline in this paper. Note that the above two models do not directly associate the topics of the citing document with the topics of the cited documents.

Nallapati and Cohn (2008) proposed an extension of Link-LDA called Link-PLSA-LDA, which is another baseline in this paper. Assuming that the citing and cited documents share similar topics, they explicitly model the information flow from the citing documents to the cited documents. In Link-PLSA-LDA, the link graph is converted into a bipartite graph in which links are connected from citing documents to cited documents. If a document has both inlinks and outlinks, it will be duplicated on both sides of the bipartite graph. The generative process for the citing documents is similar to that of Link-LDA, while the cited documents have a different generative process.

Dietz et al (2007) proposed a topic model for citation analysis. Their goal is to find topical influence of publications in research communities. They convert the citation graph (created from the publications) into a bipartite graph as in Link-PLSA-LDA. The content of a citing document is assumed to be generated by a mixture over the topic distribution of the citing document and the topic distributions of the cited documents. The differences between the topic distributions of citing and cited documents are measured, and the cited documents which have the strongest influence on the citing document are identified.

Note that in most existing models described above the hyperlinks are assumed to be generated and link prediction is an important task, while in the HTM model in this paper, the hyperlinks are assumed to be given in advance, and the key task is topic identification. In the existing models for hypertexts, the content of a document (the word distribution of the document) are not decided by the other documents. In contrast, in HTM, the content of a document is determined by itself as well as its cited documents. Furthermore, HTM is a generative model which can generate the contents of all the hypertexts in a collection, given the link structure of the collection. Therefore, if the goal is to accurately learn and pre-

Table 1: Notations and explanations.

T	Number of topics
$\mathbf{D}$	Documents in corpus
D	Number of documents
$\alpha_\theta, \alpha_\beta$	Hyperparameters for θ and β
λ	Hyperparameter to control the weight between the citing document and the cited documents
θ	Topic distributions for all documents
β	Word distribution for topic
b, c, z	Hidden variables for generating word
d	document (index)
$\mathbf{w}_d$	Word sequence in document d
N_d	Number of words in document d
L_d	Number of documents cited by document d
I_d	Set of cited documents for document d
i_{dl}	Index of l^{th} cited document of document d
ξ_d	Distribution on cited documents of document d
θ_d	Topic distribution associated with document d
b_{dn}	Decision on way of generating n^{th} word in document d
c_{dn}	Cited document that generates n^{th} word in document d
z_{dn}	Topic of n^{th} word in document d

Algorithm 1 Generative Process of HTM

```
for each document $d$ do
  Draw $\theta_d \sim Dir(\alpha_\theta)$.
end for
for each word $w_{dn}$ do
  if $L_d > 0$ then
    Draw $b_{dn} \sim Ber(\lambda)$
    Draw $c_{dn} \sim Uni(\xi_d)$
    if $b_{dn} = 1$ then
      Draw $z_{dn} \sim Multi(\theta_d)$
    else
      Draw $z_{dn} \sim Multi(\theta_{I_{dc_{dn}}})$
    end if
  else
    Draw a topic $z_{dn} \sim Multi(\theta_d)$
  end if
  Draw a word $w_{dn} \sim P(w_{dn} \mid z_{dn}, \beta)$
end for
```

dict contents of documents, the use of HTM seems more reasonable.

3 Hypertext Topic Model

3.1 Model

In topic modeling, a probability distribution of words is employed for a given document. Specifically, the probability distribution is defined as a mixture over latent topics, while each topic is future characterized by a distribution of words (Hofmann, 1999; Blei et al., 2003). In this paper, we introduce an extension of LDA model for hypertexts. Table 1 gives the major notations and their explanations.

The graphic representation of conventional LDA is given in Figure 1(a). The generative process of LDA has three steps. Specifically, in each document a topic distribution is sampled from a prior distribution defined as Dirichlet distribution. Next, a topic is sampled from the topic distribution of the document, which is a multinominal distribution. Finally, a word is sampled according to the word distribution of the topic, which also forms a multinormal distribution.

The graphic representation of HTM is given in Figure 1(b). The generative process of HTM is described in Algorithm 1. First, a topic distribution is sampled for each document according to Dirichlet distribution. Next, for generating a word in a document, it is decided whether to use the current document or documents which the document cites. (The weight between the citing document and cited documents is controlled by an adjustable hyperparameter λ.) It is also determined which cited document to use (if it is to use cited documents). Then, a topic is sampled from the topic distribution of the selected document. Finally, a word is sampled according to the word distribution of the topic. HTM naturally mimics the process of writing a hypertext document by humans (repeating the processes of writing native texts and anchor texts).

The formal definition of HTM is given below. Hypertext document d has N_d words $\mathbf{w_d} = w_{d1} \cdots w_{dN_d}$ and L_d cited documents $I_d = \{i_{d1}, \ldots, i_{dL_d}\}$. The topic distribution of d is θ_d and topic distributions of the cited documents are $\theta_i, i \in I_d$. Given λ, θ, and β, the conditional probability distribution of $\mathbf{w_d}$ is defined as:

$$p(\mathbf{w_d}|\lambda, \theta, \beta) = \prod_{n=1}^{N_d} \sum_{b_{dn}} p(b_{dn}|\lambda) \sum_{c_{dn}} p(c_{dn}|\xi_d)$$

$$\sum_{z_{dn}} p(z_{dn}|\theta_d)^{b_{dn}} p(z_{dn}|\theta_{i_{dc_{dn}}})^{1-b_{dn}} p(w_{dn}|z_{dn}, \beta).$$

Here ξ_d, b_{dn}, c_{dn}, and z_{dn} are hidden variables. When generating a word w_{dn}, b_{dn} determines whether it is from the citing document or the cited documents. c_{dn} determines which cited document it is when $b_{dn} = 0$. In this paper, for simplicity we assume that the cited documents are equally likely to be selected, i.e., $\xi_{di} = \frac{1}{L_d}$.

Note that θ represents the topic distributions of all the documents. For any d, its word distribution is affected by both θ_d and $\theta_i, i \in I_d$. There is a propagation of topics from the cited documents to the citing document through the use of $\theta_i, i \in I_d$.

For a hypertext document d that does not have cited documents. The conditional probability distribution degenerates to LDA:

$$p(\mathbf{w_d}|\theta_d, \beta) = \prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_{dn}|\theta_d) p(w_{dn}|z_{dn}, \beta).$$

By taking the product of the marginal probabilities of hypertext documents, we obtain the conditional probability of the corpus $\mathbf{D}$ given the hyperparameters $\lambda, \alpha_\theta, \beta$,

$$\begin{aligned}
&p(\mathbf{D}|\lambda, \alpha_\theta, \beta) = \\
&\int \prod_{d=1}^{D} p(\theta_d|\alpha_\theta) \prod_{n=1}^{N_d} \sum_{b_{dn}} p(b_{dn}|\lambda) \sum_{c_{dn}} p(c_{dn}|\xi_d) \\
&\sum_{z_{dn}} p(z_{dn}|\theta_d)^{b_{dn}} p(z_{dn}|\theta_{I_{dc_{dn}}})^{1-b_{dn}} \\
&p(w_{dn}|z_{dn}, \beta) d\theta. \qquad (1)
\end{aligned}$$

Note that the probability function (1) also covers the special cases in which documents do not have cited documents.

In HTM, the content of a document is decided by the topics of the document as well as the topics of the documents which the document cites. As a result contents of documents can be 'propagated' along the hyperlinks. For example, suppose web page A cites page B and page B cites page C, then the content of page A is influenced by that of page B, and the content of page B is further influenced by the content of page C. Therefore, HTM is able to more accurately represent the contents of hypertexts, and thus is more useful for text processing such as topic discovery and document classification.

3.2 Inference and Learning

An exact inference of the posterior probability of HTM may be intractable, we employ the mean field variational inference method (Wainwright and Jordan, 2003; Jordan et al., 1999) to conduct approximation. Let $I[\cdot]$ be an indicator function. We first define the following factorized variational posterior distribution q with respect to the corpus:

$$q = \prod_{d=1}^{D} q(\theta_d|\gamma_d)$$

$$\prod_{n=1}^{N_d} \left(q(x_{dn}|\rho_{dn})(q(c_{dn}|\psi_{dn}) \right)^{I[L_d>0]} q(z_{dn}|\phi_{dn}) \, ,$$

where γ, ψ, ϕ, and ρ denote free variational parameters. Parameter γ is the posterior Dirichlet parameter corresponding to the representations of documents in the topic simplex. Parameters ψ, ϕ, and ρ correspond to the posterior distributions of their associated random variables. We then minimize the KL divergence between q and the true posterior probability of the corpus by taking derivatives of the loss function with respect to variational parameters. The solution is listed as below.

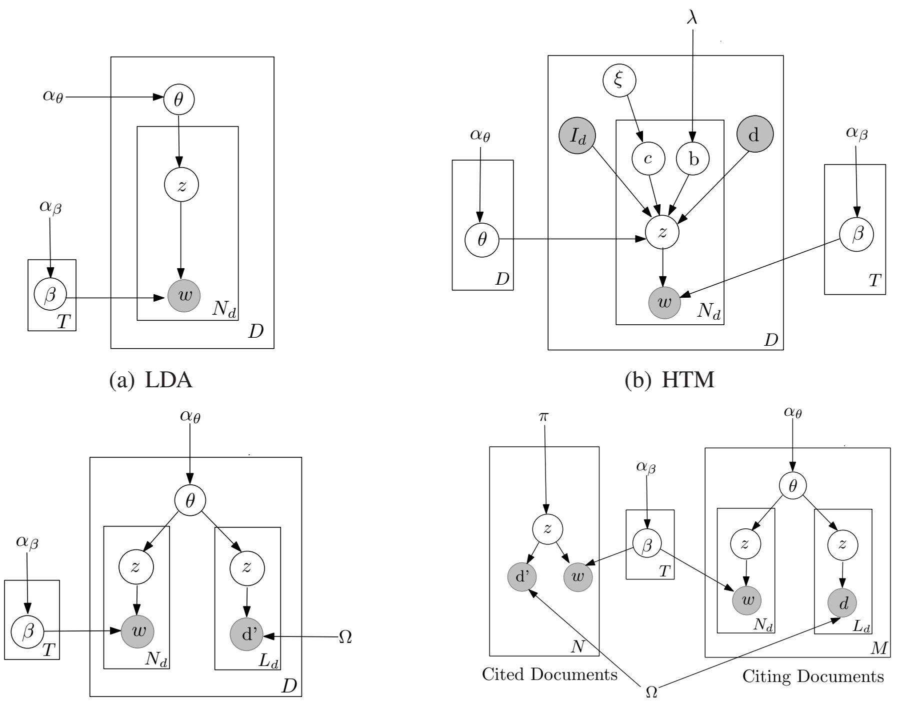
Let β_{iv} be $p(w_{dn}^v = 1|z^i = 1)$ for the word v. If $L_d > 0$, we have

E-step:

$$\begin{aligned}
\gamma_{d_i} &= \alpha_{\theta_i} + \sum_{n=1}^{N_d} \rho_{dn}\phi_{dni} + \sum_{d'=1}^{D} \sum_{l=1}^{L_{d'}} I\left[i_{d'l} = d\right] \\
&\quad \sum_{n=1}^{N_{d'}} (1 - \rho_{d'n})\psi_{d'nl}\phi_{d'ni} \, .
\end{aligned}$$

$$\begin{aligned}
\phi_{dni} &\propto \beta_{iv} \exp\left\{\rho_{dn} E_q\left[\log\left(\theta_{di}\right)|\gamma_d\right]\right. \\
&+ \left.(1 - \rho_{dn}) \sum_{l=1}^{L_d} \psi_{dnl} E_q\left[\log\left(\theta_{I_{dl}i}\right)|\gamma_{I_{dl}}\right]\right\} .
\end{aligned}$$

$$\begin{aligned}
\rho_{dn} &= \Bigg(1 + \Big(\exp\Big\{\sum_{i=1}^{k} \big((\phi_{dni} E_q[\log(\theta_{di})|\gamma_d] \\
&- \sum_{l=1}^{L_d} \psi_{dnl}\phi_{dni} E_q[\log(\theta_{I_{dl}i})|\gamma_{I_{dl}}]\big) \\
&+ \log\lambda - \log\left(1 - \lambda\right)\Big\}\Big)^{-1}\Bigg)^{-1} .
\end{aligned}$$



(a) LDA (b) HTM (c) Link-LDA (d) Link-PLSA-LDA

Figure 1: Graphical model representations

$$\varphi_{dnl} \propto \xi_{dl} \exp\{(1-\rho_{dn}) \sum_{i=1}^{k} \phi_{dni} E_q[\log(\theta_{I_{dl}i})|\gamma_{I_{dl}}]\}.$$

Otherwise,

$$\gamma_{d_i} = \alpha_{\theta_i} + \sum_{n=1}^{N_d} \phi_{dni} + \sum_{d'=1}^{D} \sum_{l=1}^{L_{d'}} I\left[i_{d'l} = d\right] \sum_{n=1}^{N_{d'}} (1-\rho_{d'n}) \psi_{d'nl} \phi_{d'ni} \, .$$

$$\phi_{dni} \propto \beta_{iv} \exp\left\{E_q\left[\log\left(\theta_{di}\right)|\gamma_d\right]\right\}.$$

From the first two equations we can see that the cited documents and the citing document jointly affect the distribution of the words in the citing document.

M-step:

$$\beta_{ij} \propto \sum_{d=1}^{D} \sum_{n=1}^{N_d} \phi_{dni} w_{dn}^{j}.$$

In order to cope with the data sparseness problem due to large vocabulary, we employ the same technique as that in (Blei et al., 2003). To be specific, we treat β as a $K * V$ random matrix, with each row being independently drawn from a Dirichlet distribution $\beta_i \sim Dir(\alpha_\beta)$. Variational inference is modified appropriately.

4 Experimental Results

We compared the performances of HTM and three baseline models: LDA, Link-LDA, and Link-PLSA-LDA in topic discovery and document classification. Note that LDA does not consider the use of link information; we included it here for reference.

4.1 Datasets

We made use of three datasets. The documents in the datasets were processed by using the Lemur Took kit (http://www.lemurproject.org), and the low frequency words in the datasets were removed.

The first dataset WebKB (available at http://www.cs.cmu.edu/˜webkb) contains six subjects (categories). There are 3,921 documents and 7,359 links. The vocabulary size is 5,019.

The second dataset Wikipedia (available at http://www.mpi-inf.mpg.de/˜angelova) contains four subjects (categories): Biology, Physics, Chemistry, and Mathematics. There are 2,970 documents and 45,818 links. The vocabulary size is 3,287.

The third dataset is ODP composed of homepages of researchers and their first level outlinked pages (cited documents). We randomly selected five subjects from the ODP archive. They are Cognitive Science (CogSci), Theory, NeuralNetwork (NN), Robotics, and Statistics. There are 3,679 pages and 2,872 links. The vocabulary size is 3,529.

WebKB and Wikipedia are public datasets widely used in topic model studies. ODP was collected by us in this work.

4.2 Topic Discovery

We created four topic models HTM, LDA, Link-LDA, and Link-PLSA-LDA using all the data in each of the three datasets, and evaluated the topics obtained in the models. We heuristically set the numbers of topics as 10 for ODP, 12 for WebKB, and 8 for Wikipedia (i.e., two times of the number of true subjects). We found that overall HTM can construct more understandable topics than the other models. Figure 2 shows the topics related to the subjects created by the four models from the ODP dataset. HTM model can more accurately extract the three topics: Theory, Statistic, and NN than the other models. Both LDA and Link-LDA had mixed topics, labeled as 'Mixed' in Figure 2. Link-PLSA-LDA missed the topic of Statistics. Interestingly, all the four models split Cognitive Science into two topics (showed as CogSci-1 and CogSci-2), probably because the topic itself is diverse.

4.3 Document Classification

We applied the four models in the three datasets to document classification. Specifically, we used the word distributions of documents created by the models as feature vectors of the documents and used the subjects in the datasets as categories. We further randomly divided each dataset into three parts (training, validation, and test) and conducted 3-fold cross-validation experiments. In each trial, we trained an SVM classifier with the training data, chose parameters with the validation data, and conducted evaluation on classification with the test data. For HTM, we chose the best λ value with the validation set in each trial. Table 2 shows the classification accuracies. We can see that HTM performs better than the other models in all three datasets.

Table 2: Classification accuracies in 3-fold cross-validation.

	LDA	HTM	Link-LDA	Link-PLSA-LDA
ODP	0.640	**0.698**	0.535	0.581
WebKB	0.786	**0.795**	0.775	0.774
Wikipedia	0.845	**0.866**	0.853	0.855

Table 3: Sign-test results between HTM and the three baseline models.

	LDA	Link-LDA	Link-PLSA-LDA
ODP	0.0237	2.15e-05	0.000287
WebKB	0.0235	0.0114	0.00903
Wikipedia	1.79e-05	0.00341	0.00424

We conducted sign-tests on all the results of the datasets. In most cases HTM performs statistically significantly better than LDA, Link-LDA, and Link-PLSA-LDA (p-value < 0.05). The test results are shown in Table 3.

4.4 Discussion

We conducted analysis on the results to see why HTM can work better. Figure 3 shows an example homepage from the ODP dataset, where superscripts denote the indexes of outlinked pages. The homepage contains several topics, including Theory, Neural network, Statistics, and others, while the cited pages contain detailed information about the topics. Table 4 shows the topics identified by the four models for the homepage. We can see that HTM can really more accurately identify topics than the other models.

The major reason for the better performance by HTM seems to be that it can fully leverage the infor-

Table 4: Comparison of topics identified by the four models for the example homepage. Only topics with probabilities > 0.1 and related to the subjects are shown.

Model	Topics	Probabilities
LDA	Mixed	0.537
HTM	Theory	0.229
	NN	0.278
	Statistics	0.241
Link-LDA	Statistics	0.281
Link-PLSA-LDA	Theory	0.527
	CogSci-2	0.175

(a) LDA

Mixed	NN	Robot	CogSci-1	CogSci-2
statistic	learn	robot	visual	conscious
compute	conference	project	model	psychology
algorithm	system	file	experiment	language
theory	neural	software	change	cognitive
complex	network	code	function	experience
mathematics	model	program	response	brain
model	international	data	process	theory
science	compute	motor	data	philosophy
computation	ieee	read	move	science
problem	proceedings	start	observe	online
random	process	build	perception	mind
analysis	computation	comment	effect	concept
paper	machine	post	figure	physical
method	science	line	temporal	problem
journal	artificial	include	sensory	content

(b) HTM

Theory	Statistics	NN	Robot	CogSci-1	CogSci-2
compute	model	learn	robot	conscious	memory
science	statistic	system	project	visual	psychology
algorithm	data	network	software	experience	language
theory	experiment	neural	motor	change	science
complex	sample	conference	sensor	perception	cognitive
computation	process	model	code	move	brain
mathematics	method	compute	program	theory	human
paper	analysis	ieee	build	online	neuroscience
problem	response	international	line	physical	journal
lecture	figure	proceedings	board	concept	society
random	result	machine	read	problem	trauma
journal	temporal	process	power	philosophy	press
bound	probable	computation	type	object	learn
graph	observe	artificial	comment	content	abuse
proceedings	test	intelligence	post	view	associate

(c) Link-LDA

Statistics	Mixed	Robot	CogSci-1	CogSci-2
statistic	compute	robot	visual	conscious
model	conference	project	model	psychology
data	system	software	experiment	cognitive
analysis	learn	file	change	language
method	network	motor	function	brain
learn	computation	robotics	vision	science
sample	proceedings	informatik	process	memory
algorithm	neural	program	perception	theory
process	ieee	build	move	philosophy
bayesian	algorithm	board	response	press
application	international	sensor	temporal	online
random	science	power	object	neuroscience
distribution	complex	code	observe	journal
simulate	theory	format	sensory	human
mathematics	journal	control	figure	mind

(d) Link-PLSA-LDA

Theory	NN	Robot	CogSci-1	CogSci-2
compute	conference	robot	conscious	model
algorithm	learn	code	experience	process
computation	science	project	language	visual
theory	international	typeof	book	data
complex	system	motor	change	experiment
science	compute	control	make	function
mathematics	network	system	problem	learn
network	artificial	serve	brain	neural
paper	ieee	power	world	system
journal	intelligence	program	read	perception
proceedings	robot	software	case	represent
random	technology	file	than	vision
system	proceedings	build	mind	response
problem	machine	pagetracker	theory	object
lecture	neural	robotics	content	abstract

Figure 2: Topics identified by four models

Radford M.Neal

Professor, Dept. of Statistics and Dept. of Computer Science, University of Toronto

I'm currently highlighting the following :

∗ A new R function for performing univariate slice sampling.[1]

∗ A workshop paper on Computing Likelihood Functions for High-Energy Physics Experiments when Distributions are Defined by Simulators with Nuisance Parameters.[2]

∗ Slides from a talk at the Third Workshop on Monte Carlo Methods on "Short-Cut MCMC: An Alternative to Adaptation", May 2007: Postscript, PDF.

Courses I'm teaching in Fall 2008 :

∗ STA 437: Methods for Multivariate Data[3]

∗ STA 3000: Advanced Theory of Statistics[4]

You can also find information on courses I've taught in the past.[5]

You can also get to information on :

∗ Research interests[6] (with pointers to publications)

∗ Current and former graduate students[7]

∗ Current and former postdocs[8]

∗ Curriculum Vitae: PostScript, or PDF.

∗ Full publications list[9]

∗ How to contact me[10]

∗ Links to various places[11]

If you know what you want already, you may wish to go directly to :

∗ Software available on-line[12]

∗ Papers available on-line[13]

∗ Slides from talks[14]

∗ Miscellaneous other stuff[15]

Information in this hierarchy was last updated 2008-06-20.

Figure 3: An example homepage: http://www.cs.utoronto.ca/˜ radford/

Table 5: Word assignment in the example homepage.

Word	b_{dn}	c_{dn}	Topic	Probability
mcmc	0.544	2	Stat	0.949
experiment	0.546	2	Stat	0.956
neal	0.547	8	NN	0.985
likelihood	0.550	2	Stat	0.905
sample	0.557	2	Stat	0.946
statistic	0.559	2	Stat	0.888
parameter	0.563	2	Stat	0.917
perform	0.565	2	Stat	0.908
carlo	0.568	2	Stat	0.813
monte	0.570	2	Stat	0.802
toronto	0.572	8	NN	0.969
distribution	0.578	2	Stat	0.888
slice	0.581	2	Stat	0.957
energy	0.581	13	NN	0.866
adaptation	0.591	7	Stat	0.541
teach	0.999	11	Other	0.612
current	0.999	11	Other	0.646
curriculum	0.999	11	Other	0.698
want	0.999	11	Other	0.706
highlight	0.999	10	Other	0.786
professor	0.999	11	Other	0.764
academic	0.999	11	Other	0.810
student	0.999	11	Other	0.817
contact	0.999	11	Other	0.887
graduate	0.999	11	Other	0.901

Table 6: Most salient topics in cited pages.

URL	Topic	Probability
2	Stat	0.690
7	Stat	0.467
8	NN	0.786
13	NN	0.776

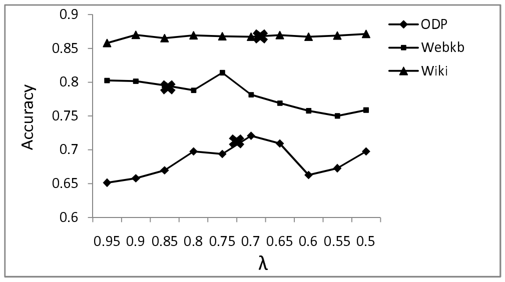


Figure 4: Classification accuracies on three datasets with different λ values. The cross marks on the curves correspond to the average values of λ in the 3-fold cross-validation experiments.

mation from the cited documents. We can see that the content of the example homepage is diverse and not very rich. It might be hard for the other baseline models to identify topics accurately. In contrast, HTM can accurately learn topics by the help of the cited documents. Specifically, if the content of a document is diverse, then words in the document are likely to be assigned into wrong topics by the existing approaches. In contrast, in HTM with propagation of topic distributions from cited documents, the words of a document can be more accurately assigned into topics. Table 5 shows the first 15 words and the last 10 words for the homepage given by HTM, in ascending order of b_{dn}, which measures the degree of influence from the cited documents on the words (the smaller the stronger). The table also gives the values of c_{dn}, indicating which cited documents have the strongest influence. Furthermore, the topics having the largest posterior probabilities for the words are also shown. We can see that the words 'experiment', 'sample', 'parameter', 'perform', and 'energy' are accurately classified. Table 6 gives the most salient topics of cited documents. It also shows the probabilities of the topics given by HTM. We can see that there is a large agreement between the most salient topics in the cited documents and the topics which are affected the most in the citing document.

Parameter λ is the only parameter in HTM which needs to be tuned. We found that the performance of HTM is not very sensitive to the values of λ, which reflects the degree of influence from the cited documents to the citing document. HTM can perform well with different λ values. Figure 4 shows the classification accuracies of HTM with respect to different λ values for the three datasets. We can see that HTM works better than the other models in most of the cases (cf., Table 2).

5 Conclusion

In this paper, we have proposed a novel topic model for hypertexts called HTM. Existing models for processing hypertexts were developed based on the assumption that both words and hyperlinks are stochastically generated by the model. The generation of latter type of data is actually unnecessary for representing *contents* of hypertexts. In the HTM model, it is assumed that the hyperlinks of hypertexts are given and only the words of the hypertexts are stochastically generated. Furthermore, the word distribution of a document is determined not only by the topics of the document in question but also from the topics of the documents which the document cites. It can be regarded as 'propagation' of topics reversely along hyperlinks in hypertexts, which can lead to more accurate representations than the existing models. HTM can naturally mimic human's process of creating a document (i.e., by considering using the topics of the document and at the same time the topics of the documents it cites). We also developed methods for learning and inferring an HTM model within the same framework as LDA (Latent Dirichlet Allocation). Experimental results show that the proposed HTM model outperforms the existing models of LDA, Link-LDA, and Link-PLSA-LDA on three datasets for topic discovery and document classification.

As future work, we plan to compare the HTM model with other existing models, to develop learning and inference methods for handling extremely large-scale data sets, and to combine the current method with a keyphrase extraction method for extracting keyphrases from web pages.

6 Acknowledgement

We thank Eric Xing for his valuable comments on this work.

References

Ricardo Baeza-Yates and Berthier Ribeiro-Neto. 1999. Modern Information Retrieval. *ACM Press / Addison-Wesley*.

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

David Blei and John Lafferty. 2005. Correlated Topic Models. In *Advances in Neural Information Processing Systems 12*.

David Blei and John Lafferty. 2006. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*.

Chaitanya Chemudugunta, Padhraic Smyth, and Mark Steyvers. 2007. Modeling General and Specific Aspects of Documents with a Probabilistic Topic Model. In *Advances in Neural Information Processing Systems 19*.

David Cohn and Huan Chang. 2000. Learning to Probabilistically Identify Authoritative Documents. In *Proceedings of the 17rd international conference on Machine learning*.

David Cohn and Thomas Hofmann. 2001. The missing link - a probabilistic model of document content and hypertext connectivity. In *Neural Information Processing Systems 13*.

Laura Dietz, Steffen Bickel and Tobias Scheffer. 2007. Unsupervised prediction of citation influences. In *Proceedings of the 24th international conference on Machine learning*.

Elena Erosheva, Stephen Fienberg, and John Lafferty. 2004. Mixed-membership models of scientific publications. In *Proceedings of the National Academy of Sciences*, 101:5220–5227.

Thomas Griffiths and Mark Steyvers. 2004. Finding Scientific Topics. In *Proceedings of the National Academy of Sciences, 101 (suppl. 1)* .

Thomas Griffiths, Mark Steyvers, David Blei, and Joshua Tenenbaum. 2005. Integrating Topics and Syntax. In *Advances in Neural Information Processing Systems, 17*.

Amit Gruber, Michal Rosen-Zvi, and Yair Weiss. 2008. Latent Topic Models for Hypertext. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*.

Thomas Hofmann. 1999. Probabilistic Latent Semantic Analysis. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*.

Michael Jordan, Zoubin Ghahramani, Tommy Jaakkola, and Lawrence Saul. 1999. An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37(2):183–233.

QiaoZhu Mei, Deng Cai, Duo Zhang, and ChengXiang Zhai. 2008. Topic Modeling with Network Regularization. In *Proceeding of the 17th international conference on World Wide Web*.

Thomas Minka and John Lafferty. 2002. Expectation-Propagation for the Generative Aspect Model. In *Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence*.

Ramesh Nallapati and William Cohen. 2008. Link-PLSA-LDA: A new unsupervised model for topics and influence of blogs. In *International Conference for Webblogs and Social Media*.

Martin Wainwright, and Michael Jordan. 2003. Graphical models, exponential families, and variational inference. In *UC Berkeley, Dept. of Statistics, Technical Report, 2003*.

Xing Wei and Bruce Croft. 2006. LDA-based document models for ad-hoc retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval.*

A Japanese Predicate Argument Structure Analysis using Decision Lists

Hirotoshi Taira, Sanae Fujita, Masaaki Nagata
NTT Communication Science Laboratories
2-4, Hikaridai, Seika-cho,
Keihanna Science City,
Kyoto 619-0237, Japan
{{taira,sanae}@cslab.kecl, nagata.masaaki@lab}.ntt.co.jp

Abstract

This paper describes a new automatic method for Japanese predicate argument structure analysis. The method learns relevant features to assign case roles to the argument of the target predicate using the features of the words located closest to the target predicate under various constraints such as dependency types, words, semantic categories, parts of speech, functional words and predicate voices. We constructed decision lists in which these features were sorted by their learned weights. Using our method, we integrated the tasks of semantic role labeling and zero-pronoun identification, and achieved a 17% improvement compared with a baseline method in a sentence level performance analysis.

1 Introduction

Recently, predicate argument structure analysis has attracted the attention of researchers because this information can increase the precision of text processing tasks, such as machine translation, information extraction (Hirschman et al., 1999), question answering (Narayanan and Harabagiu, 2004) (Shen and Lapata, 2007), and summarization (Melli et al., 2005). In English predicate argument structure analysis, large corpora such as FrameNet (Fillmore et al., 2001), PropBank (Palmer et al., 2005) and NomBank (Meyers et al., 2004) have been created and utilized. Recently, the GDA Corpus (Hashida, 2005), Kyoto Text Corpus Ver.4.0 (Kawahara et al., 2002) and NAIST Text Corpus (Iida et al., 2007) were constructed in Japanese, and these corpora have become the target of an automatic Japanese predicate argument structure analysis system. We conducted Japanese predicate argument structure (PAS) analysis for the NAIST Text Corpus, which is the largest of these three corpora, and, as far as we know, this is the first time PAS analysis has been conducted for whole articles of the corpus.

The NAIST Text Corpus has the following characteristics, i) semantic roles for both predicates and event nouns are annotated in the corpus, ii) three major case roles,[1] namely the ga, wo and ni-cases in Japanese are annotated for the base form of predicates and event nouns, iii) both the case roles in sentences containing the target predicates and those outside the sentences (zero-pronouns) are annotated, and iv) coreference relations are also annotated.

As regards i), recently there has been an increase in the number of papers dealing with nominalized predicates (Pradhan et al., 2004) (Jiang and Ng, 2006) (Xue, 2006) (Liu and Ng, 2007). For example, 'trip' in the sentence "During my trip to Italy, I met him." refers not only to the event "I met him" but also to the event "I traveled to Italy." As in this example, nouns sometimes have argument structures referring to an event. Such nouns are called *event nouns* (Komachi et al., 2007) in the NAIST Text Corpus. At the same time, the problems related to compound nouns are also important. In Japanese, a compound noun sometimes simultaneously contains both an event noun and its arguments. For example, the compound noun, '企業買収 (corporate buyout)' contains an event noun '買収 (buyout)' and its accusative, '企業 (corporate).' However, compound

[1]Kyoto Text Corpus has about 15 case roles.

nouns provide no information about syntactic dependency or about case markers, so it is difficult to specify the predicate-argument structure. Komachi et al. investigated the argument structure of event nouns using the co-occurrence of target nouns and their case roles in the same sentence (Komachi et al., 2007). In these approaches, predicates and event nouns are dealt with separately. Here, we try to unify these different argument structures using decision lists.

As regards ii), for example, in the causative sentence, 'メアリーはトムに夕食を作らせる (Mary makes Tom fix dinner),' the basic form of the causative verb, '作らせる (make fix)' is '作る (fix),' and its nominative is 'トム (Tom)' and the accusative case role (wo-case) is '夕食 (dinner),' although the surface case particle is ni (dative). We must deal with syntactic transformations in passive, causative, and benefactive constructions when analyzing the corpus.

As regards iii) and iv), in Japanese, zero pronouns often occur, especially when the argument has already been mentioned in previous sentences. There have been many studies of zero-pronoun identification (Walker et al., 1994) (Nakaiwa, 1997) (Iida et al., 2006).

In this paper, we present a general procedure for handling both the case role assignment of predicates and event nouns, and zero-pronoun identification. We use the decision list learning of rules to find the closest words with various constraints, because with decision lists the readability of learned lists is high and the learning is fast.

The rest of this paper is organized as follows. We describe the NAIST Text Corpus, which is our target corpus in Section 2. We describe our proposed method in Section 3. The result of experiments using the NAIST Text Corpus and our method are reported in Section 4 and our conclusions are provided in Section 5.

2 NAIST Text Corpus

In the NAIST Text Corpus, three major obligatory Japanese case roles are annotated, namely the ga-case (nominative or subjective case), the wo-case (accusative or direct object) and the ni-case (dative or in-direct object). The NAIST Text Corpus is based on the Kyoto Text Corpus Ver. 3.0, which contains 38,384 sentences in 2,929 texts taken from news articles and editorials in a Japanese newspaper, the 'Mainichi Shinbun'.

We divided these case roles into four types by location in the article as in (Iida et al., 2006), i) the case role depends on the predicate or the predicate depends on the case role in the intra-sentence ('dependency relations'), ii) the case role does not depend on the predicate and the predicate does not depend on the case role in the intra-sentence ('zero-anaphoric (intra-sentential)'), iii) the case role is not in the sentence containing the predicate ('zero-anaphoric (inter-sentential)'), and iv) the case role and the predicate are in the same phrase ('in same phrase'). Here, we do not deal with exophora.

We show the distribution of the above four types in test samples in our split of the NAIST Text Corpus in Tables 1 and 2. In predicates, the 'dependency relations' type in the wo-case and the ni-case occur frequently. In event nouns, the 'zero-anaphoric (intra-sentential)' and 'zero-anaphoric (inter-sentential)' types in the ga-case occur frequently. With respect to the 'in same phrase' type, the wo-case occurs frequently.

3 Predicate Argument Structure Analysis using Features of Closest Words

In this section, we describe our algorithm. In the algorithm, we used various constraints when searching for the words located closest to the target predicate. We described these constraints as features with the direct products of dependency types (ic, oc, ga_c, wo_c, ni_c, sc, nc, fw and bw), generalization levels (words, semantic categories, parts of speech), functional words and voices.

3.1 Dependency Types

In Japanese, the functional words in a phrase (*Bunsetsu* in Japanese) and the interdependency of bunsetsu phrases are important for determining the predicate argument structure. In accordance with the character of the dependency between the case roles and the predicates or event nouns, we divided Japanese word dependency into the following seven types that cover all dependency types in Japanese. Additionally, we use two optional dependency types.

Table 1: Distribution of case roles for predicates (Test Data)

	predicate					
	ga (Nominative)		wo (Accusative)		ni (Dative)	
all	15,996	(100.00%)	8,348	(100.00%)	4,871	(100.00%)
dependency relations	9,591	(59.96%)	7,184	(86.06%)	4,276	(87.78%)
zero-anaphoric (intra-sentential)	3,856	(24.11%)	870	(10.42%)	360	(7.39%)
zero-anaphoric (inter-sentential)	2,496	(15.60%)	225	(2.70%)	132	(2.71%)
in same phrase	53	(0.33%)	69	(0.83%)	103	(2.11%)

Table 2: Distribution of case roles for event nouns (Test Data)

	event noun					
	ga (Nominative)		wo (Accusative)		ni (Dative)	
all	4,099	(100.00%)	2,314	(100.00%)	423	(100.00%)
dependency relations	977	(23.84%)	648	(28.00%)	105	(24.82%)
zero-anaphoric (intra-sentential)	1,672	(40.79%)	348	(15.04%)	135	(31.91%)
zero-anaphoric (inter-sentential)	1,040	(25.37%)	165	(7.13%)	44	(10.40%)
in same phrase	410	(10.00%)	1,153	(49.83%)	139	(32.86%)

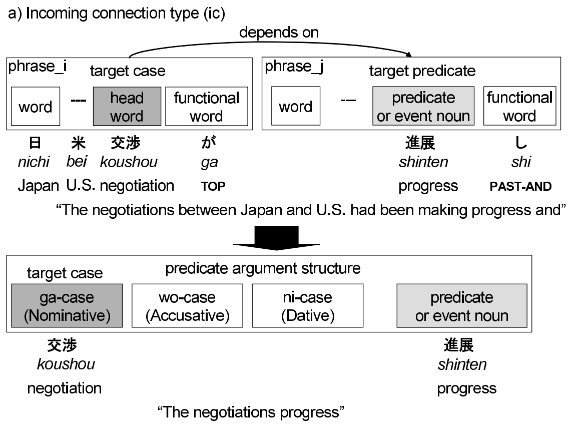


Figure 1: Type ic

3.1.1 Incoming Connection Type (ic)

With this type, the target case role is the headword of a bunsetsu phrase and the case role phrase depends on the target predicate phrase (Figure 1).

3.1.2 Outgoing Connection Type (oc)

With this type, the target case role is the headword of a phrase and a phrase containing a target predicate or event noun depends on the case role phrase (Figure 2).

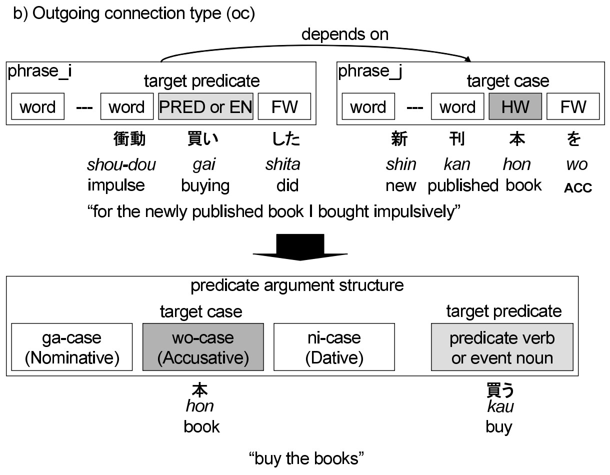

b) Outgoing connection type (oc)
depends on
phrase_i
target predicate
word
---
word
PRED or EN
FW
phrase_j
target case
word
---
word
HW
FW
衝動 買い した 新 刊 本 を
shou-dou gai shita shin kan hon wo
impulse buying did new published book ACC
"for the newly published book I bought impulsively"
predicate argument structure
target case
target predicate
ga-case (Nominative)
wo-case (Accusative)
ni-case (Dative)
predicate verb or event noun
本 hon book
買う kau buy
"buy the books"

Figure 2: Type oc

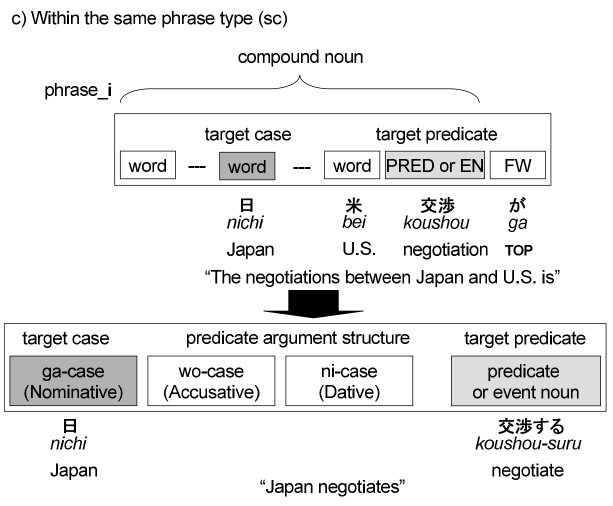


Figure 3: Type sc

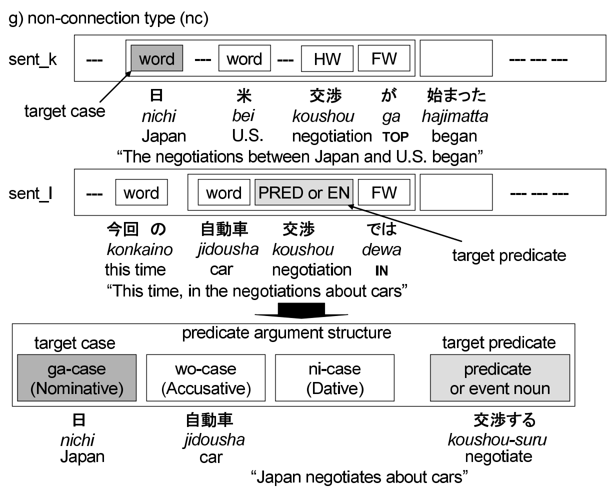


Figure 5: Type nc

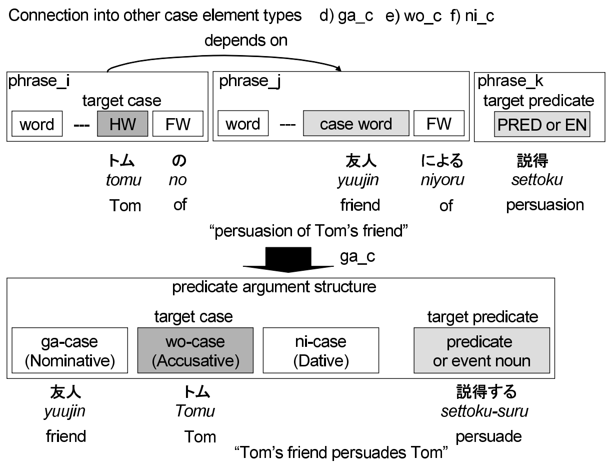


Figure 4: Type ga_c, wo_c, ni_c

3.1.3 'Within the Same Phrase' Type (sc)

With this type, the target case role and the target predicate or event noun are in the same phrase (Figure 3).

3.1.4 'Connection into Other Case role Types (ga_c, wo_c, ni_c)

With these types, a phrase containing the target case role depends on a phrase containing another predetermined case role (Figure 4). We use the terms 'ga_c', 'wo_c' and 'ni_c' when the predetermined case roles are the ga-case, wo-case and ni-case, respectively.

3.1.5 Non-connection Type (nc)

With this type, a phrase containing the target case role and a phrase containing the target predicate or event noun are in the same article, but these phrases do not depend on each other (Figure 5).

3.1.6 Optional Type (fw and bw)

Type fw and bw stand for 'forward' and 'backward' types, respectively. Type fw means the word located closest to the target predicate or event noun without considering functional words or voices. With fw, the word is located between the top of the article containing the target predicate and the target predicate or event noun. Similarly, type bw means the word located closest to the target predicate or noun, which is located between the targeted predicate or event noun, and the tail of the article containing the predicate.

3.2 Generalization Levels

We used three levels of generalization for every case role candidate, that is, word, semantic category, and part of speech. Every word is annotated with a part of speech in the Kyoto Text Corpus, and we used these annotations. With regard to semantic categories, we annotated every word with a semantic category based on a Japanese thesaurus, Nihongo Goi Taikei. The thesaurus consists of a hierarchy of 2,710 semantic classes, defined for over 264,312 nouns, with a maximum depth of twelve (Ikehara et al., 1997). We mainly used the semantic classes of

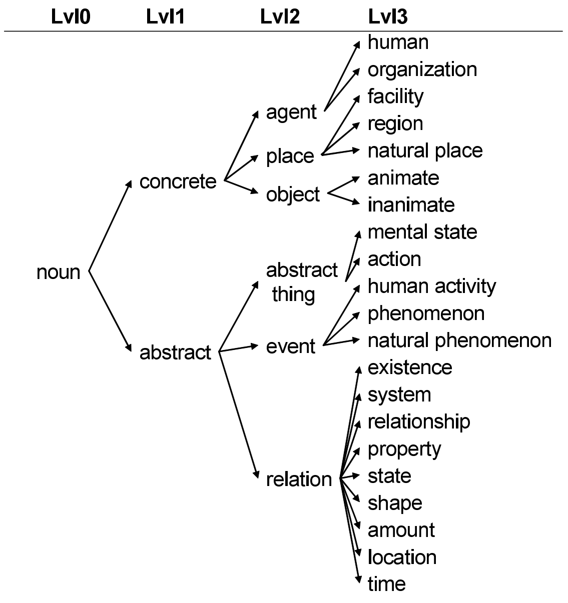


Figure 6: Top 3 levels of the Japanese thesaurus, 'Nihongo Goi Taikei'

the third level, and partly the fourth level, which are similar to semantic roles. We show the top three levels of the Nihongo Goi Taikei common noun thesaurus in Figure 6. We annotated the words with their semantic category by hand.

3.3 Functional Word and Voice

We used a functional word in the phrase containing the target case role and active and passive voices for the predicate as base features.

3.4 Training Algorithm

The training algorithm used for our method is shown in Figure 7. First, the algorithm constructs features that search for the words located closest to the target predicate under various constraints. Next, the algorithm learns by using linear Support Vector Machines (SVMs) (Vapnik, 1995). SVMs learn effective features by the one vs. rest method for every case role. We used TinySVM [2] as an SVM implementation. Moreover, we construct decision lists sorted by weight from linear SVMs. Finally, the algorithm calculates the existing probabilities of case roles for every predicate or event noun. This step produces the criterion that decides whether or not we will determine the case roles when there is no interdependency between the case role candidate and the predicate.

Our split of the NAIST Text Corpus has only 62,264 training samples for 2,874 predicates, and we predict that there will be a shortage of training samples when adopting traditional learning algorithms, such as learning algorithms using entropy. So, we used SVMs with a high generalization capability to learn the decision lists.

3.5 Test Algorithm

The test algorithm of our method is shown in Figure 8. In the test phase, we analyzed test samples using decision lists and the existing probabilities of case roles learned in the training phase. In step 1, we determined case roles using a decision list consisting of features exhibiting case role and predicate interdependency, that is, ic, oc, ga_c, wo_c, and ni_c. This is because there are many cases in Japanese where the syntactic constraint is stronger than the semantic constraint when we determine the case roles. In step 2, we determined case roles using a decision list of sc ('in same phrase') for the case roles that were not determined in step 1. This step was mainly for event nouns. Japanese event nouns frequently form compound nouns that contain case roles. In step 3, we decided whether or not to proceed to the next step by using the existing probabilities of case roles. If the probability was less than a certain threshold (50%), then the algorithm stopped. In step 4, we determined case roles using a decision list of the features that have no interdependency, that is, nc, fw and bw. This step will be executed when the target case role is syntactically necessary and determined by the co-occurrence of the case roles and predicate or event noun without syntactic clues, such as dependency, functional words and voices.

4 Experimental Results

4.1 Experimental Setting

We performed our experiments using the NAIST Text Corpus 1.4β (Iida et al., 2007). We used 49,527 predicates and 12,737 event nouns from articles published from January 1st to January 11th and the editorials from January to August as training ex-

[2]http://chasen.org/taku/software/TinySVM/

```
for each predicate $p_i$ in all predicates appeared in the training corpus do
  $feature\_list(p_i) = \{\}$ ; $n \leftarrow 0$
  clear ($x$, $y$)
  for each instance $p_{ij}$ of $p_i$, in the training corpus do
    Clear $order()$ for all features
    $a_{ij} \leftarrow$ the article including $p_{ij}$
    $W_{ij} \leftarrow$ the number of words in $a_{ij}$
    $pred\_index \leftarrow$ the word index of $p_{ij}$ in $a_{ij}$
    for ($m = pred\_index - 1$; $m \geq 1$; $m - -$) do
      $n + +$
      $dep\_type = \mathbf{get\_dependency\_type}(w_m, p_{ij})$
      if $dep\_type$ == 'ic', 'nc', 'ga_c', 'wo_c' or 'ni_c' then inc_order($n$, $dep\_type$, $w_m$, $p_{ij}$)
      else if $dep\_type$ == 'sc' then inc_order($n$, $dep\_type$, '', '')
      endif
      inc_order($n$, 'fw', '', '')
      if $w_m$ is the ga-case role then $y_{n,ga} \leftarrow 1$ else $y_{n,ga} \leftarrow 0$
      if $w_m$ is the wo-case role then $y_{n,wo} \leftarrow 1$ else $y_{n,wo} \leftarrow 0$
      if $w_m$ is the ni-case role then $y_{n,ni} \leftarrow 1$ else $y_{n,ni} \leftarrow 0$
    end for
    for ($m = pred\_index + 1$; $m \leq W_{ij}$; $m + +$) do
      $n + +$
      $dep\_type = \mathbf{get\_dependency\_type}(w_m, p_{ij})$
      if $dep\_type$ == 'oc', 'nc', 'ga_c', 'wo_c' or 'ni_c' then inc_order($n, dep\_type, w_m, p_{ij}$)
      else if $dep\_type$ == 'sc' then inc_order($n$, $dep\_type$, '', '')
      endif
      inc_order($n$, 'bw', '', '')
      if $w_m$ is the ga-case role then $y_{n,ga} \leftarrow 1$ else $y_{n,ga} \leftarrow 0$
      if $w_m$ is the wo-case role then $y_{n,wo} \leftarrow 1$ else $y_{n,wo} \leftarrow 0$
      if $w_m$ is the ni-case role then $y_{n,ni} \leftarrow 1$ else $y_{n,ni} \leftarrow 0$
    end for
  end for
  Learn linear SVMs using ($x_1$, $y_{1,ga}$), ..., ($x_n$, $y_{n,ga}$)
  Learn linear SVMs using ($x_1$, $y_{1,wo}$), ..., ($x_n$, $y_{n,wo}$)
  Learn linear SVMs using ($x_1$, $y_{1,ni}$), ..., ($x_n$, $y_{n,ni}$)
  Make the decision list for $p_i$, sorting features by weight.
  Calculate the existing probabilities of case roles for $p_i$.
end for
procedure   get_dependency_type($w_m$, $p_{ij}$)
  if $phrase(w_m)$ depends on $phrase(p_{ij})$ then return 'ic'
  else if $phrase(p_{ij})$ depends on $phrase(w_m)$ then return 'oc'
  else if $phrase(w_m)$ depends on $phrase(p_{ga})$ then return 'ga_c'
  else if $phrase(w_m)$ depends on $phrase(p_{wo})$ then return 'wo_c'
  else if $phrase(w_m)$ depends on $phrase(p_{ni})$ then return 'ni_c'
  else if $phrase(w_m)$ equals $phrase(p_{ij})$ then return 'sc'
  else return 'nc'
end procedure
procedure   inc_order($n, dep\_type, func, voice$)
Set a feature $f_w = (w_m, dep\_type, func, voice)$ ; $order(f_w)$++ ; if $order(f_w)$ == 1 then $x_{n,f_w} \leftarrow 1$
Set a feature $f_s = (sem(w_m), dep\_type, func, voice)$ ; $order(f_s)$++ ; if $order(f_s)$ == 1 then $x_{n,f_s} \leftarrow 1$
Set a feature $f_p = (pos(w_m), dep\_type, func, voice)$ ; $order(f_p)$++ ; if $order(f_p)$ == 1 then $x_{n,f_p} \leftarrow 1$
$feature\_list(p_i) \leftarrow feature\_list(p_i) \bigcup \{f_w, f_s, f_p\}$
end procedure
```

Figure 7: Training algorithm

```
Step 1. Determine case roles using a decision list concerning ic, oc, ga_c, wo_c and ni_c.
Step 2. Determine case roles using a decision list concerning sc for undetermined case roles in
Step.1.
Step 3. If the existing probability of case roles < 50 % then the program ends.
Step 4. Determine case roles using a decision list concerning nc, fw and bw types.
```

Figure 8: Test algorithm

amples. We used 11,023 predicates and 3,161 event nouns from articles published on January 12th and 13th and the September editorials as development examples. And we used 19,501 predicate and 5,276 event nouns from articles dated January 14th to 17th and editorials dated October to December as test examples. This is a typical way to split the data.

We used the annotations in the Kyoto Text Corpus as the interdependency of bunsetsu phrases. We used both individual and multiple words as case roles. We used the phrase boundaries annotated in the NAIST Text Corpus in the training phase, and used those annotated automatically by our system using POSs and simple rules in the test phase. The accuracy of the automatic annotation is about 90%.

4.2 Baseline Method

To evaluate our algorithm, we conducted experiments using a baseline method. With the method, we used only nouns that depended on predicates or event nouns as case role candidates. If the functional word (post-positional case) in the phrase is 'ga','wo' and 'ni', we determined the ga-case, wo-case, or ni-case for the candidates. Next, as regards event nouns in compound nouns, if there was another word in a compound noun containing an event noun and it co-occurred with the event noun as a case role with a higher probability in the training samples, then the word was selected for the case role.

4.3 Entropy Method

The conventional approach for making decision lists utilizes the entropy of samples selected by the rules (Yarowsky, 1994) (Goodman, 2002). We performed comparative experiments using Yarowsky's entropy algorithm (Yarowsky, 1994).

Table 3: Existing probabilities of case roles for predicates and event nouns

Predicate	Existing Probability		
or Event Noun	ga (NOM)	wo (ACC)	ni (DAT)
使う (use)	44.72%	82.92%	5.33%
交渉 (negotiation)	77.41%	30.70%	0.00%
参加 (participation)	87.09%	0.00%	72.46%
基づく (based on)	81.89%	0.00%	100.00%

4.4 Overall Results

The overall results are shown in Table 7. Here, 'entropy' indicates Yarowsky's algorithm, which uses entropy (Yarowsky, 1994). Throughout the test data, the F-measure (%) of our method exceeded that of the baseline system and the 'entropy' system. With the ga-case (nominative) in particular, the F-measure increased 9 points.

Table 3 shows some examples of the existing probabilities of case roles for predicates or event nouns. When the probabilities are extreme values such as the ni-case (dative) of 交渉 (negotiation), the wo-case (accusative) of 参加 (participation), and the wo-case and ni-base of 基づく (based on), we can decide to fill the targeted case role or not with high precision. However, it is difficult to decide to fill the targeted case role or not when the probability is close to 50 percent as in the ga-case of 使う (use).

We show the learned decision list of the ic type (the case role depends on the predicate or event noun), sc type (in the same phrase) and the other types for event noun 交渉 (negotiation) in Tables 4, 5 and 6, respectively. Here, 'word' in the 'level' column means 'base form of predicate' and 'sem' means 'semantic category of predicate.' In the ic and sc type decision lists, features with semantic categories, such as 'REGION', 'LOCATION' and 'EVENT', occupy a higher order. In contrast, in the list of the other types, the features that occupy the higher order are the features of the word base

Table 4: Decision list for ic type of event noun 交渉 (negotiation)

order	case	dep_type	level	head word	functional word	voice	weight
1	ga	ic	word	北朝鮮人民共和国 (North Korea)	の (of)	active	0.9820
2	ga	ic	sem	地域 (REGION)	の (of)	active	0.6381
3	ga	ic	word	日米両国 (both Japan and U.S.)	の (of)	active	0.5502
4	wo	ic	word	合弁会社設立 (establishment of joint ventures)	の (of)	active	0.5288
5	wo	ic	word	電気通信分野 (telecommunications)	の (of)	active	0.4142
6	wo	ic	word	北朝鮮人民共和国 (North Korea)	との (for)	active	0.3168
7	wo	ic	word	行為 (ACTION)	の (of)	active	0.3083
8	ga	ic	sem	未分類語 (OOV NOUN)	の (of)	active	0.2939
9	wo	ic	word	自動車・同部品分野 (car and auto parts sector)	の (of)	active	0.2775
10	wo	ic	sem	場 (LOCATION)	の (of)	active	0.2471

Table 5: Decision list for sc type of event noun 交渉 (negotiation)

order	case	dep_type	level	head word	weight
1	wo	sc	sem	事象 (EVENT)	1.1738
2	wo	sc	word	協定 (arrangement)	1.0000
3	ga	sc	word	日中航空 (airline of Japan and China)	0.9392
4	wo	sc	sem	思考 (MENTAL STATE)	0.8958
5	ga	sc	word	日米金融サービス分野 (financial services of Japan and U.S.)	0.8371
6	wo	sc	word	契約更改 (contract extension)	0.7870
7	wo	sc	word	合弁 (joint venture)	0.7865
8	wo	sc	word	知的所有権 (intellectual property rights)	0.7224
9	wo	sc	word	自動車・同部品 (car and auto parts)	0.7196
10	ga	sc	word	日朝 (Japan and North Korea)	0.6771

Table 6: Decision list for other types of event noun 交渉 (negotiation)

order	case	dep_type	level	head word	functional word	voice	weight
1	ga	fw	word	日米 (Japan and U.S.)			1.9954
2	ga	fw	word	台湾 (Taiwan)			1.9952
3	ga	fw	word	米朝 (U.S. and North Korea)			1.4979
4	ga	fw	word	英中 (U.K. and China)			1.1773
5	ga	nc	word	両国 (both nations)	は (TOP)	active	1.1379
6	wo	fw	word	国交正常化 (diplomatic normalization)			1.0000
7	ga	bw	word	米朝 (U.S. and North Korea)			1.0000
8	ga	fw	word	労使 (capital and labor)			1.0000
9	wo	fw	word	自動車分野 (automotive area)			1.0000
10	ga	nc	word	双方 (both sides)	は (TOP)	active	1.0000

Table 7: Overall results for NAIST Text Corpus (F-measure(%))

	training data				test data			
	sentence	ga (NOM)	wo (ACC)	ni (DAT)	sentence	ga (NOM)	wo (ACC)	ni (DAT)
baseline	25.32	32.58	74.51	82.70	21.34	30.08	69.48	76.62
entropy	**73.46**	**89.53**	**92.72**	91.09	33.10	45.67	73.28	77.77
our method	64.81	86.76	92.52	**92.20**	**38.06**	**55.07**	**75.82**	**80.45**

Table 8: Results for predicates in test sets (F-measure(%))

	baseline / our method		
	ga (Nominative)	wo (Accusative)	ni (Dative)
all	34.44 / **57.40**	77.00 / **79.50**	79.83 / **83.15**
dependency relations	51.96 / **75.53**	85.42 / **88.20**	81.83 / **89.51**
zero-anaphoric (intra-sentential)	0.00 / **30.15**	0.00 / **11.41**	0.00 / **3.66**
zero-anaphoric (inter-sentential)	1.85 / **23.45**	3.00 / **9.32**	0.00 / **11.76**
in same phrase	0.00 / **75.00**	0.00 / **51.78**	0.00 / **84.65**

Table 9: Results for event nouns (F-measure(%))

	baseline / our method		
	ga (Nominative)	wo (Accusative)	ni (Dative)
all	11.05 / **45.64**	32.30 / **61.80**	20.85 / **38.88**
dependency relations	12.98 / **68.01**	25.00 / **62.46**	40.00 / **56.05**
zero-anaphoric (intra-sentential)	0.00 / **36.19**	0.00 / **20.46**	0.00 / **6.62**
zero-anaphoric (inter-sentential)	1.40 / **23.25**	1.06 / **10.37**	0.00 / **3.51**
in same phrase	58.76 / **78.93**	47.44 / **77.96**	28.91 / **58.13**

form. This means local knowledge of relations between case roles and predicates or event nouns in the word level is more important than semantic level knowledge.

4.5 Results for Predicates in Test Sets

We show the results we obtained for predicates in Table 8. The results reveal that our method is superior to the baseline system. Our algorithm is particularly effective in the ga-case.

4.6 Results for Event Nouns in Test Sets

We show the results we obtained for event nouns in Table 9. This also shows that our method is superior to the baseline system. The precision with sc type is high and our method is effective as regards event nouns.

5 Conclusion

We presented a new method for Japanese automatic predicate argument structure analysis using decision lists based on the features of the words located closest to the target predicate under various constraints. The method learns the relative weights of these different features for case roles and ranks them using decision lists. Using our method, we integrated the knowledge of case role determination and zero-pronoun identification, and generally achieved a high precision in Japanese PAS analysis. In particular, we can extract knowledge at various levels from the corpus for event nouns. In future, we will use richer constraints and research better ways of distinguishing whether or not cases are obligatory.

Acknowledgments

We thank Ryu Iida and Yuji Matsumoto of NAIST for the definitions of the case roles in the NAIST Text Corpus and functional words, and Franklin Chang for valuable comments.

References

Charles J. Fillmore, Charles Wooters, and Collin F. Baker. 2001. Building a large lexical databank which provides deep semantics. In *Proc. of the Pacific Asian Conference on Language, Information and Computation (PACLING)*.

Joshua Goodman. 2002. An incremental decision list learner. In *Proc. of the ACL-02 Conference on Empirical Methods in Natural Language Processing(EMNLP02)*, pages 17–24.

Kouichi Hashida. 2005. Global document annotation (GDA) manual. http://i-content.org/GDA/.

Lynette Hirschman, Patricia Robinson, Lisa Ferro, Nancy Chinchor, Erica Brown, Ralph Grishman, and Beth Sundheim. 1999. Hub-4 Event'99 general guidelines.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2006. Exploiting syntactic patterns as clues in zero-anaphora resolution. In *Proc. of the 21st International Confer-*

ence on Computational Linguistics and 44th Annual Meeting of the ACL, pages 625–632.

Ryu Iida, Mamoru Komachi, Kentaro Inui, and Yuji Matsumoto. 2007. Annotating a Japanese text corpus with predicate-argument and coreference relations. In *Proc. of ACL 2007 Workshop on Linguistic Annotation*, pages 132–139.

Satoru Ikehara, Masahiro Miyazaki, Satoshi Shirai, Akio Yokoo, Hiromi Nakaiwa, Kentaro Ogura, Yoshifumi Ooyama, and Yoshihiko Hayashi. 1997. *Nihongo Goi Taikei, A Japanese Lexicon*. Iwanami Shoten, Tokyo.

Zheng Ping Jiang and Hwee Tou Ng. 2006. Semantic role labeling of NomBank: A maximum entropy approach. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*.

Daisuke Kawahara, Sadao Kurohashi, and Koichi Hashida. 2002. Construction of a Japanese relevance-tagged corpus (in Japanese). *Proc. of the 8th Annual Meeting of the Association for Natural Language Processing*, pages 495–498.

Mamoru Komachi, Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2007. Learning-based argument structure analysis of event-nouns in Japanese. In *Proc. of the Conference of the Pacific Association for Computational Linguistics (PACLING)*, pages 120–128.

Chang Liu and Hwee Tou Ng. 2007. Learning predictive structures for semantic role labeling of NomBank. In *Proc. of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 208–215.

Gabor Melli, Yang Wang, Yudong Liu, Mehdi M. Kashani, Zhongmin Shi, Baohua Gu, Anoop Sarkar, and Fred Popowich. 2005. Description of SQUASH, the SFU question answering summary handler for the DUC-2005 summarization task. In *Proc. of DUC 2005*.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The NomBank project: An interim report. In *Proc. of HLT-NAACL 2004 Workshop on Frontiers in Corpus Annotation*.

Hiromi Nakaiwa. 1997. Automatic identification of zero pronouns and their antecedents within aligned sentence pairs. In *Proc. of the 3rd Annual Meeting of the Association for Natural Language Processing (in Japanese)*.

Srini Narayanan and Sanda Harabagiu. 2004. Question answering based on semantic structures. In *Proc. of the 20th International Conference on Computational Linguistics (COLING)*.

M. Palmer, P. Kingsbury, and D. Gildea. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Sameer Pradhan, Waybe Ward, Kadri Hacioglu, James Martin, and Dan Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proc. of the Human Language Technology Conference/North American Chapter of the Association of Computational Linguistics HLT/NAACL 2004*.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Proc. of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, pages 12–21.

V. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.

M. Walker, M. Iida, and S. Cote. 1994. Japanese discourse and the process of centering. *Computational Linguistics*, 20(2):193–233.

Nianwen Xue. 2006. Semantic role labeling of nominalized predicates in Chinese. In *Proc. of the HLT-NAACL*, pages 431–438.

David Yarowsky. 1994. Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French. In *Proc. of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 88–95.

Online Word Games for Semantic Data Collection

David Vickrey Aaron Bronzan William Choi Aman Kumar
Jason Turner-Maier Arthur Wang Daphne Koller
Stanford University
Stanford, CA 94305-9010
{dvickrey,abronzan,aman,arthurex,koller}@cs.stanford.edu
{wchoi25,jasonptm}@stanford.edu

Abstract

Obtaining labeled data is a significant obstacle for many NLP tasks. Recently, online games have been proposed as a new way of obtaining labeled data; games attract users by being fun to play. In this paper, we consider the application of this idea to collecting semantic relations between words, such as hypernym/hyponym relationships. We built three online games, inspired by the real-life games of Scattergories^{TM} and Taboo^{TM}. As of June 2008, players have entered nearly 800,000 data instances, in two categories. The first type of data consists of category/answer pairs ("Types of vehicle","car"), while the second is essentially free association data ("submarine","underwater"). We analyze both types of data in detail and discuss potential uses of the data. We show that we can extract from our data set a significant number of new hypernym/hyponym pairs not already found in WordNet.

1 Introduction

One of the main difficulties in natural language processing is the lack of labeled data. Typically, obtaining labeled data requires hiring human annotators. Recently, building online games has been suggested an alternative to hiring annotators. For example, von Ahn and Dabbish (2004) built the ESP Game[1], an online game in which players tag images with words that describe them. It is well known that there are large numbers of web users who will play online games. If a game is fun, there is a good chance that sufficiently many online users will play.

We have several objectives in this paper. The first is to discuss design decisions in building word games for collecting data, and the effects of these decisions. The second is to describe the word games that we implemented and the kinds of data they are designed to collect. As of June 2008, our games have been online for nearly a year, and have collected nearly 800,000 data instances. The third goal is to analyze the resulting data and demonstrate that the data collected from our games is potentially useful in linguistic applications. As an example application, we show that the data we have collected can be used to augment WordNet (Fellbaum, 1998) with a significant number of new hypernyms.

[1]www.gwap.com/gwap/gamesPreview/espgame

2 General Design Guidelines

Our primary goal is to produce a large amount of clean, useful data. Each of these three objectives ("large", "clean", and "useful") has important implications for the design of our games.

First, in order to collect large amounts of data, the game must be attractive to users. If the game is not fun, people will not play it. This requirement is perhaps the most significant factor to take into account when designing a game. For one thing, it tends to discourage extremely complicated labeling tasks, since these are more likely to be viewed as work. It would certainly be a challenge (although not necessarily impossible) to design a game that yields labeled parse data, for example.

In this paper, we assume that if people play a game in real life, there is a good chance they will play it online as well. To this end, we built online versions of two popular "real-world" games: Scattergories^{TM} and Taboo^{TM}. Not only are these games fun, but there is also a preexisting demand for online versions of these games, driving search traffic to our site. We will go into more detail about these games in the next section.

An important characteristic of these games is that they involve more than one player. Interacting with another player increases the sense of fun. Another important feature these games share is that they are timed. Timing has several advantages. First, timing helps make the games feel more "game-like", by adding a sense of urgency. Without timing, it risks feeling more like a labeling task than a game.

The next requirement is that the data be clean. First, the players must be capable of producing high-quality annotations. Second, the game should encourage users to enter relevant data. We award points as a motivating factor, but this can lead players to enter irrelevant data, or collude with other players, in order to get a higher score. In particular, collusion is more likely when players can freely communicate. An excellent technique for producing good data, used effectively in the ESP game, is to require the players to match on their inputs. Requiring players to match their partner's hidden answers discourages off-topic answers and makes it quite difficult to collude (requiring outside communication). We use this technique in all of our games.

Finally, the data must be useful. Ideally, it would be directly applicable to an NLP task. This requirement can come into conflict with the other goals. There are certainly many kinds of data that would be useful for NLP tasks (such as labeled parses), but designing a game to collect this data that people will play and that produces clean data is difficult.

In this paper, we focus on a particular kind of linguistic data: semantic relationships between pairs of words and/or phrases. We do this for several reasons. First, this kind of data is relatively simple, leading to fun games which produce relatively clean data. Second, the real-world games we chose to emulate naturally produce this kind of data. Third, there are a number of recent works which focus on extracting these kinds of relationships, e.g. (Snow et al., 2006; Nakov & Hearst, 2008). Our work presents an interesting new way of extracting this type of data. Finally, at least one of these kinds of relationships, the hypernym, or "X is a Y" relation, has proven to be useful for a variety of NLP tasks.

3 Description of Our Games

We now describe our three games in detail.

3.1 Categorilla

Categorilla, inspired by Scattergories™, asks players to supply words or phrases which fit specific categories, such as "Things that fly" or "Types of fish". In addition, each game has a specific letter which all answers must begin with. Thus, if the current game has letter "b", reasonable answers would be "bird" and "barracuda", respectively. In each game, a randomly matched pair of players are given the same 10 categories; they receive points when they match with the other player for a particular category. Players are allowed to type as may answers for a given category as they wish (until a match is made for that category). After a match is made, the players get to see what word they matched on for that category. Each answer is supposed to fit into a specific category, so the data is automatically structured.

Our system contains 8 types of categories, many of which were designed to correspond to linguistic resources used in NLP applications. Table 1 describes the category types.

The purpose of the first three types of categories is to extract hypernym/hyponym pairs like those found in WordNet (e.g., "food" is a hypernym of "pizza"). In fact, the categories were automatically generated from WordNet, as follows. First, we assigned counts C_s to each synset s in WordNet using the SemCor[2] labeled data set of word senses. Let $desc(s)$ be the set of descendants of s in the hypernym hierarchy. Then for each pair of synsets s, d, where $d \in desc(s)$, we computed a conditional distribution $P(d|s) = \frac{C_d}{\sum_{d' \in desc(s)} C_{d'}}$, the probability that we choose node d from among the descendants of s. Finally, we computed the entropy of each node s in WordNet, $\sum_{d \in desc(s)} P(d|s) log P(d|s)$. Synsets with many different descendants occurring in SemCor will have higher entropies. Each node with a sufficiently high entropy was chosen as a category.

We then turned each synset into a category by taking the first word in that synset and plugging it into one of several set phrases. For nouns, we tried two variants ("Types of food" and "Foods"). Depending on the noun, either of these may be more natural (consider "Cities" vs. "Types of city"). "Types of food" tends to produce more adjectival answers than "Foods". We tried only one variation for verbs ("Methods of paying"). This phrasing is not perfect; in particular, it encourages non-verb answers like "credit card".

The second group of categories tries to capture selectional preferences of verbs – for example, "ba-

[2]Available at `www.cs.unt.edu/ rada/downloads.html`

Name	#	Description	Example	Good Answer
NHyp	269	Members of a class of nouns	“Vehicles”	“car”
NType	269	Members of a class of nouns	“Types of vehicle”	“car”
VHyp	70	Members of a class of verbs	“Methods of cutting”	“trimming”
VS	1380	Subjects of a verb	“Things that eat”	“cats”
VO	909	Direct objects of a verb	“Things that are abandoned”	“family”
VPP	77	Preposition arguments of a verb	“Things that are accused of”	“crime”
Adj	219	Things described by an adjective	“Things that are recycled”	“cans”
O	105	Other; mostly “Things found at/in ...”	“Things found in a school”	“teachers”

Table 1: Summary of category types. # indicates the number of categories of that type.

nana” makes sense as the object of “eat” but not as the subject. Our goal with these categories was to produce data useful for automatically labeling semantic roles (Gildea & Jurafsky, 2002), where selectional preferences play an important role. We tried three different types of categories, corresponding to subjects, objects, and prepositional objects. Examples are “Things that eat”, “Things that are eaten”, and “Things that are eaten with”, to which good answers would be “animals”, “food”, and “forks”. These categories were automatically generated using the labeled parses in Penn Treebank (Marcus et al., 1993) and the labeled semantic roles of PropBank (Kingsbury et al., 2002). To generate the object categories, for example, for each verb we then counted the number of times a core argument (ARG0-ARG5) appeared as the direct object of that verb (according to the gold-standard parses), and used all verbs with count at least 5. This guaranteed that all generated categories were grammatically correct and captured information about core arguments for that verb. Most of the prepositional object categories proved to be quite confusing (e.g., “Things that are acted as”), so we manually removed all but the most clear. Not surprisingly, the use of the Wall Street Journal had a noticeable effect on the types of categories extracted; they have a definite financial bias.

The third group of categories only has one type, which consists of adjective categories such as “Things that are large”. While we did not have any specific task in mind for this category type, having a database of attributes/noun pairs seems potentially useful for various NLP tasks. To generate these categories, we simply took the most common adjectives in the SemCor data set. Again, the resulting set of adjectives reflect the corpus; for example, “Things that are green” was not generated as a category, while “Things that are corporate” was.

The final group of categories were hand-written. This group was added to make sure that a sufficient number of “fun” categories were included, since some of the category types, particularly the verb categories, are somewhat confusing and difficult. Most of the hand-written categories are of the form “Things found at/in X”, where X is a location, such as “Japan” or “the ocean”.

The starting letter requirement also has important consequences for data collection. It was designed to increase the variety of obtained data; without this restriction, players might produce a smaller set of “obvious” answers. As we will see in the results, this restriction did indeed lead to a great diversity of answers, but at a severe cost to data quality.

3.2 Categodzilla

Categodzilla is a slightly modified version of Categorilla, with the starting letter constraint relaxed. The combination of difficult categories and rare letters often leads to bad answers in Categorilla. To increase data quality, in Categodzilla for each category there are three boxes. In the first box you can type any word you want. Answers in the second box must start with a given “easy” letter such as “c”. Answers in the third box must start with a given “hard” letter, such as “k”. The boxes much be matched in order; guesses typed in the first box which match either of the other two boxes are automatically propagated.

3.3 Free Association

Free Association, inspired by Taboo^{TM}, simply asks players to type words related to a given “seed” word. Players are not allowed to type any of several words on a “taboo” list, specific to the current seed word.

As soon as a match is achieved, players move on to a new seed word.

The seed words came from two sources. The first was the most common words in SemCor. The second was the Google unigram data, which lists the most common words on the web. In both cases, we filtered out stop words (including all prepositions).

Unlike Categorilla, we found that nearly all collected Free Association data was of good quality, due to the considerably easier nature of the task. Of course, we do lose the structure present in Categorilla. As the name suggests, the collected data is essentially free word association pairs. We analyze the data in depth to see what kinds of relations we got.

4 Existing Word Games

Two notable word games already exist for collecting linguistic data. The first is the Open Mind Common Sense system[3] (Chklovski, 2003). The second is Verbosity[4] (von Ahn et al., 2006). Both these games are designed to extract common sense facts, and thus have a different focus than our games.

5 Bots

There may not always be enough players available online to match a human player with another human player. Therefore, one important part of designing an online game is building a bot which can function in the place of a player. The bots for all of our games are similar. Each has a simple random model which determines how long to wait between guesses. The bot's guesses are drawn from past guesses made by human players for that category/seed word (plus starting letter in the case of Categorilla). Just as with a human player, as soon as one of the bot's guesses matches one of the player's, a match is made.

If there are no past guesses, the bot instead makes "imaginary" guesses. For example, in Categorilla, we make the (obviously false) assumption that for every category and every starting letter there are exactly 20 possible answers, and that both the player's guesses and the bot's imaginary guesses are drawn from those 20 answers. Then, given the number of guesses made by the player and the number of imaginary guesses made by the bot, the probability of a match can be computed (assuming that all guesses are made independently). Once this probability passes a certain threshold, randomly generated for each category at the start of each game, the bot matches one of the player's guesses, chosen at random. The Free Association bot works similarly.

[3]http://commons.media.mit.edu/en

[4]www.gwap.com/gwap/gamesPreview/verbosity

	Grla	Gdza	Free
Game Length	3min	3min	2min
Games Played	19656	2999	15660
Human-Human Games	428	45	401
Categories	3298	3298	9488
Guesses Collected	391804	78653	307963
Guesses/Categories	119	24	32
Unique Guesses	340433	56142	221874
Guesses: All/Unique	1.15	1.40	1.39
Guesses/Games	19.9	26.2	19.7
Guesses per minute	6.6	8.7	9.9

Table 2: Statistics for Categorilla, Categodzilla, and Free Association.

For Free Association, the bot rarely has to resort to generating these imaginary guesses. In Categorilla, due to the starting letter requirement, the bot has to make imaginary guesses much more often. Imaginary guessing can encourage poor behavior on the part of players, since they see that matches can occur for obviously bad answers. They may also realize that they are playing against a bot.

An additional complication for Categorilla and Categodzilla is that the bot has to decide which categories to make guesses for, and in what order. Our current guessing model takes into account past difficulty of the category and the current guessing of the human player to determine where to guess next.

6 Users and Usage

Table 2 shows statistics of each of the games, as of late June 2008. While we have collected nearly 800,000 data instances, nearly all of the games were between a human and the bot. Over the course of a year, our site received between 40 and 100 visits a day; this was not enough to make it likely for human-human games to occur. The fact that we still collected this amount of data suggests that our bot is a satisfactory substitue for a human teammate. We have anecdotally found that most players do not realize they are playing against a bot. While most of the data comes from games between a human and a bot, our data set consists only of input by the human players.

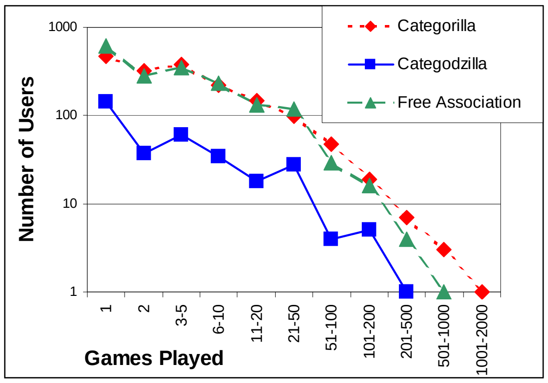


Figure 1: Users are grouped by number of games played. Note that this graph is on a double-log scale.

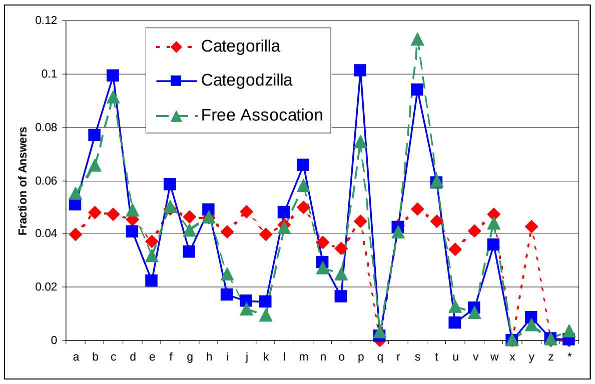


Figure 2: Fraction of answers with given initial letter. * denotes everything nonalphabetical.

Our main tool for attracting traffic to our site was Google. First, we obtained $1 a day in AdWords, which pays for between 7 to 10 clicks on our ad a day. Second, our site is in the top 10 results for many relevant searches, such as “free online scattergories”.

Categorilla was the most popular of the games, with about 25% more games played than Free Association. Taking the longer length of Categorilla games into account (see Table 2), this corresponds to almost 90% more play time. This is despite the fact that Free Association is the first game listed on our home page. We hypothesize that this is because Scattergories^{TM} is a more popular game in real life, and so many people come to our site specifically looking for an online Scattergories^{TM} game. Categodzilla has been played signficantly less; it has been available for less time and is listed third on the site. Even for Categodzilla, the least played game, we have collected on average 24 guesses per category.

Several of our design decisions for the games were based on trying to increase the diversity of answers. Categorilla has the highest answer diversity. For a given category, each answer occurred on average only 1.15 times. In general, this average should increase with the amount of collected data. However, Categodzilla and Free Association have collected significantly fewer answers per category than Categorilla, but still have a higher average, around 1.4. The high answer diversity of Categorilla is a direct result of the initial letter constraint. For all three games, the majority of category/answer pairs occurred only once.

Figure 1 shows the distribution over users of the number of games played. Not surprisingly, it follows the standard Zipfian curve; there are a large number of users who have played only a few games, and a few users who have played a lot of games. The middle of the curve is quite thick; for both Categorilla and Free Association there are more than 100 players who have played between 21 and 50 games.

Figure 2 shows the distribution of initial letters of collected answers for each game. Categorilla is nearly flat over all letters besides ’q’, ’x’, and ’z’ which are never chosen as the inital letter constraint. This means players make a similar number of guesses even for difficult initial letters. In contrast, the distribution of initial letters for Free Association data reflects the relatively frequency of initial letters in English. Even though Categodzilla does have letter constraints in the 2nd and 3rd columns, its statistics over initial letter are very similar to Free Association.

7 Categorilla and Categodzilla Data

In our analyses, we take ALL guesses made at any time, whether or not they actually produced a match. This greatly increases the amount of usable data, but also increases the amount of noise in the data.

The biggest question about the data collected from Categorilla and Categodzilla is the quality of the data. Many categories can be difficult or somewhat confusing, and the initial letter constraint further increases the difficulty.

To evaluate the quality of the data, we asked three volunteer labelers to label 1000 total category/answer pairs. Each labeler labeled every pair with one of three labels, ’y’, ’n’, or ’k’. ’y’ means that the answer fit the category. ’n’ means that it

Annotator	y	k	n
#1	72	13	115
#2	77	27	96
#3	88	42	70
Majority	76	29	95

Table 3: Comparison of annotators

Data Set	y	k	n
Control	30	14	156
Categorilla	76	29	95
Categodzilla	144	23	33

Table 4: Overall answer accuracy

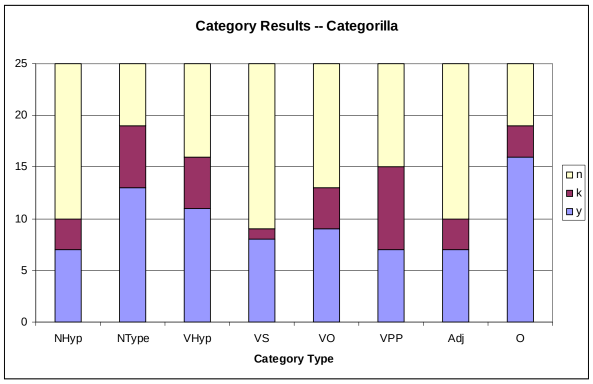


Figure 3: Categorilla accuracy by category type

does not fit. 'k' means that it "kind of" fits. This was mostly left up to the labelers; the only suggestion was that one use of 'k' could be if the category was "Things that eat" and the answer was "sandwich." Here, the answer is clearly related to the category, but doesn't actually fit.

The inter-annotator agreement was reasonable, with a Fleiss' kappa score of .49. The main difference between annotators was how permissive they were; the percentage of answers labeled 'n' ranged from 58% for the first annotator to 35% for the third. The labeled pairs were divided into 5 subgroups of 200 pairs each (described below); Table 3 shows the number of each label for the Categorilla-Random subset. We aggregated the different annotations by taking a majority vote; if all three answers were different, the item was labeled 'k'. Table 3 also shows the statistics of the majority vote on the same subset.

Overall Data Quality. We compared results for three random subsets of answers, Control-Random, Categorilla-Random, and Categodzilla-Random. Categorilla-Random was built by selecting 200 random category/answer pairs from the Categorilla data. Note that category/answer pairs that occurred more than once were more likely to be selected. Categodzilla-Random was built similarly. Control-Random was built by randomly selecting two sets of 200 category/answer pairs each (including data from both Categorilla and Categodzilla), and then combining the categories from the first set with the answers from the second to generate a set of random category/answer pairs.

Table 4 shows results for these three subsets. The chance for a control answer to be labeled 'y' was 15%. Categorilla produces data that is significantly better than control, with 38% of answers labeled 'y'. Categodzilla, which is more relaxed about initial letter restrictions, is significantly better than Categorilla, with 72% of answers labeled 'y'. This relaxation has an enormous impact on the quality of the data. Note however that these statistics are not adjusted for accuracy of individual players; it may be that only more accurate players play Categodzilla.

Effect of Category Type on Data Quality. Within each type of category (see Table 1), certain categories appear much more often than others due to the way categories are selected (at least two "easy" categories are guaranteed every game). To adjust for this, we built a subset of 200 category/answer pairs by selecting 25 different categories randomly from each type of category. We then selected an answer at random from among the answers submitted for that category. In addition, we built a control set using the same 200 categories but instead using answers selected at random from the entire Categorilla data set. Results for Categorilla data are shown in Figure 3; we omit the corresponding graph for control for lack of space. For most categories, the Categorilla data is significantly better than the control. The hand-written category type, O, has the best data quality, which is not surprising because these categories allow the most possible answers, and thus are easiest of think of answers for. These categories also have the highest number of 'y' labels for the control. Next best are the hypernym categories, NType. NType is much higher than the other noun hypernym category NHyp because the "Type of" phrasing is generally more natural and allows for adjectival answers. The VPP category type, which tries to extract prepositional objects, contains

Data Set	Letters	Size	y	k	n
Control	Easy	127	.14	.08	.78
Control	Hard	72	.15	.06	.79
Categorilla	Easy	106	.45	.14	.41
Categorilla	Hard	94	.30	.15	.55

Table 5: Accuracy of easy letters vs. hard letters. Size is the number of answers for that row.

the most number of 'k' annotations; this is because players often put answers that are subjects or objects of the verb, such as "pizza" for "Things that are eaten with". The adjective category type, Adj, has the lowest increase over the control; this is likely due to the nature of the extracted adjectives.

Effect of Initial Letter on Data Quality. In general, we would expect common initial letters to yield better data since there are more possible answers to choose from. We did not have enough labeled data to do letter by letter statistics. Instead, we broke the letters into two groups, based on the empirical difficulty of obtaining matches when given that initial letter. The easy letters were 'abcfhlmnprst', while the hard letters were 'degijkouvwy'. Table 5 shows the results on Categorilla-Random and Control-Random on these two subsets. First, note that the results on Control-Random are the same for hard letters and easy letters. This means that words starting with common letters are not more likely to fit in a category. For both hard letters and easy letters, the accuracy is considerably better on the Categorilla data. However, the increase in the number of 'y' labels for easy letters is twice that for hard letters. The quality of data for hard letters is considerably worse than that for easy letters.

8 Free Association Data

In contrast to Categorilla and even Categodzilla, we found that the Free Association data was quite clean. However, it is also not structured; we simply get pairs of related words. Thus, the essential question for this game is what kind of data we get.

To analyze the types of relationships between words, the authors labeled 500 randomly extracted unique pairs with a rich set of word-word relations, described in Table 6. This set of relations was designed to capture the observed relationships encountered in the Free Association data. Unlike our Categorilla labeled set, pairs that occurred more than once were NOT more likely to be selected than pairs that occurred once (i.e., the category/answer pairs were aggregated prior to sampling). Sampling in this way led to more diversity in the pairs extracted.

To label each pair, the authors found a sequence of relationships which connected the two words. In many cases, this was a single link. For example, "dragon" and "wing" are connected by a single link, "wing" IS PART OF "dragon". In others, multiple links were required. For the seed word "dispute" and answer "arbitrator", we can connect using two links: "dispute" IS OBJECT OF "resolve", "arbitrator" IS SUBJECT OF "resolve". There were two other possible ways to label a pair. First, they might be totally unrelated (i.e., a bad answer). Second, they might be related, but not connectable using our set of basic relations. For example, "echo" is clearly related to "valley", but in a complicated way.

The quality of the data is considerably higher than Categorilla and Categodzilla; under 10% of words are unrelated. Slightly over 20% of the pairs are labeled Misc, i.e., the words are related but in a complicated way. 3% of the pairs can be linked with a chain of two simple relations. The remaining 67% of all pairs were linked with a single simple relation.

The category Desc deserves some discussion. This category included both simple adjective descriptions, such as "creek" and "noisy", and also qualifiers, such as "epidemic" and "typhoid", where one word specifies what kind of thing the other is. The distinction between Desc and Phrase was simply based on to what extent the combination of the two words was a set phrase (such as "east" and "Germany").

Schulte im Walde et al. (2008) address very similar issues to those discussed in this section. They built a free association data set containing about 200,000 German word pairs using a combination of online and offline volunteers (but not a game). They then analyze the resulting associations by comparing the resulting pairs to a large-scale lexical resource, GermaNet (the German counterpart of WordNet). Our data analysis was by hand, making it comparatively small scale but more detailed. It would be interesting to compare the data sets to see whether the use of a game affects the resulting data.

9 Filtering Bad Data

In this section, we consider a simple heuristic for filtering bad data: only retaining answers that were

Name	#	Description	Example
Misc	103	Words related, but in a complicated way	“echo”, “valley”
Desc	76	One of the words describes the other	“cards”, “business”
None	47	Words are not related	“congress”,“store”
Syn	46	The words are synonyms	“downturn”, “dip”
Obj	33	One word is the object of the other	“exhale”,“emission”
Hyp	30	One word is an example of the other	“cabinet”,“furniture”
$\approx$Syn	29	The words are “approximate” synonyms	“maverick”,“outcast”
Cousin	21	The words share a common hypernym (is-a) relation	“meter”,“foot”
Has	18	One word “has” the other	“supermarket”,“carrots”
2-Chain	15	Words are linked by a chain of two simple relations	“arbitrator”,“dispute”
Phrase	13	Words make a phrase; similar to Desc	“East”, “Germany”
Part	11	One is a part of the other	“dragon”,“wings”
At	10	One is found at the other	“harbor”, “lake”
Subj	8	One is the subject of the other	“actor”, “pretend”
Form	7	One is a form of the other	“revere”,“reverence”
Def	7	One defines the other	“blind”,”unable to see”
Opp	7	The two are opposites	“positive”,“negative”
Sound	6	The two words sound similar	“boutique”,“antique”
Sub	5	One is a subword of the other	“outlet”, “out”
Unit	2	One is a unit of the other	“reel”,“film”
Made	2	One is made of the other	“knee”,“bone”

Table 6: Relation types for 500 hand-labeled examples. # indicates the number of pairs with that label.

guessed some minimum number of times. Note that in this section all answers were stemmed in order to combine counts across plurals and verb tenses.

For the Categorilla data, filtering out category/answer pairs that only occurred once from Categorilla-Random left a total of 64 answers (from an original 200), of which 36 were labeled ’y’ and 8 were labeled ’k’. The fraction of ’y’ labels in the reduced set is 56%, up from 38% in the original set. This gain in quality comes at the cost of losing slightly over two-thirds of the data.

For Categodzilla-Random, a similar filter left 88 (out of 200), with 79 labeled ’y’ and 7 labeled ’k’. For the hand-labeled Free Association data, applying this filter yielded a total of 123 pairs (out of an original 500), with only 2 having no relation[5]. In these two games, this filter eliminates nearly all bad data while keeping a reasonable fraction of the data.

Clearly, this filter is less effective for Categorilla than the other two games. One of the main reasons for this is that the letter constraints cause people to try to fit words starting with that letter into all categories that they even vaguely relate to, rather than thinking of words that really fit that category. Examples include {“Art supplies”,“jacket”}, {“Things found in Chicago”,“king”} and {“Things that are African”,“yak”}. Of course, we can further increase the quality of the data by making the filter more restrictive, at the cost of losing more data. For example, removing answers occuring fewer than 5 times from Categorilla-Random leaves only 8 answers (out of 200), 7 labeled ’y’ and 1 labeled ’n’.

There are other ways we could filter the data. For example, suppose we are given an outside database of pairs of words which are known to be semantically related. We could apply the following heuristic: if an answer to a particular category is similar to many other answers for that category, then that answer is likely to be a good one. Preliminary experiments using distributional similarity of words as the similarity metric suggest that this heuristic captures complimentary information to the guess frequency heuristic. We leave as future work a full integration of the two heuristics into a single improved filter.

[5]The higher fraction of lost pairs for Free Association is primarily due to the method of sampling pairs for evaluation, as discussed in Section 8.

Classified Type	#	Example
Real hypernyms	96	“equipment”,“racquet”
Compound hypernyms	32	“arrangement”,“flower”
Adjectives	25	“building”,“old”
Sort-of hypernyms	14	“vegetable”,“salad”
Not hypernyms	33	“profession”,“money”

Table 7: Breakdown of potential hypernym pairs

10 Using the Data

Categorilla and Categodzilla produce structured data which is already in a usable or nearly usable form. For example, the NHyp and NType categories produce lists of hypernyms, which could be used to augment WordNet. We looked at this particular application in some detail.

First, in order to remove noisy data, we used only Categodzilla data and removed answers which occurred only once. We took all category/answer pairs where the category was of type either NHyp or NType, and where the answer was a noun. This resulted in 1604 potential hypernym/hyponym pairs. Of these, 733 (or 46%) were already in WordNet. The remaining 871 were not found in WordNet. We then hand-labeled a random subset of 200 of the 871 to determine how many of them were real hypernym/hyponym pairs. The results are shown in Table 7. Counting compound hyponyms, nearly two-thirds of the pairs are real hypernym/hyponym pairs. These new pairs could directly augment WordNet. For example, for the word “crime”, WordNet has as hyponyms “burglary” and “fraud”. However, it doesn’t have “arson”, “homicide”, or “murder”, which are among the 871 new pairs. WordNet lists “wedding” as being an “event”, but not “birthday”.

The verb subject, object, and prepositional object categories were designed to collect data about the selectional preferences of verbs. These categories turned out to be problematic for several reasons. First, statistics about selectional preferences of verbs are not too difficult to extract from the web (although in some cases they might be somewhat noisy). Thus, the motivation for extracting this data using a game is not as apparent. Second, providing arguments of verbs out of the context of a sentence may be too difficult. For example, for the category “Things that are accumulated”, there a couple of obvious answers, such as “wealth” or “money”, but beyond these it becomes more difficult. In the context of an actual document, quite a lot of things can accumulate, but outside of that context it is difficult to think of them.

One solution to this problem would be to provide context. For example, the category “Things that accumulate in your body” is both easier to think of answers for and probably collects more useful data. However, automatically creating categories with the right level of specificity is not a trivial task; our initial experiments suggested that it is easy to generate too much context, creating an uninteresting category.

The Free Association game produces a lot of very clean data, but does not classify the relationships between the words. While a web of relationships might be useful by itself, classifying the pairs by relation type would clearly be valuable. Snow et al. (2006) and Nakov and Hearst (2008), among others, look at using a large amount of unlabeled data to classify relations between words. One issue with extracting new relations from text, for example meronyms (part-of relationships), is that they tend to occur fairly rarely. Thus, it is very easy to get a large number of spurious pairs. Using our data as a set of candidate pairs for relation extraction could greatly reduce the resulting noise. We believe that application of existing techniques to the data from the Free Association game could lead to a clean, classified set of word-word relations, but leave this as future work.

11 Discussion and Future Work

One way to extend Categorilla and Categodzilla would be to add additional types of categories. For example, a meronym category type (e.g. “Parts of a car”) would work well. Further developing the verb categories (e.g., “Things that accumulate in your body”) is another challenging but interesting direction; these categories would produce phrase-word relationships rather than word-word relationships.

Probably the most interesting direction for future work is trying to increase the complexity of the data collected from a game. There are two significant difficulties: keeping the game fun, and making sure the collected data is not too noisy. One interesting question for future research is whether different game architectures might be better suited to certain kinds of data. For example, a “telephone” style game, where players relay a phrase or sentence through some noisy channel, might be an interesting way to obtain paraphrase data.

References

Chklovski, T. (2003). Using analogy to acquire commonsense knowledge from human contributors. *Thesis*.

Fellbaum, C. (Ed.). (1998). *Wordnet: An electronic lexical database*. MIT Press.

Gildea, D., & Jurafsky, D. (2002). Automatic labeling of semantic roles. *Computational Linguistics*.

Kingsbury, P., Palmer, M., & Marcus, M. (2002). Adding semantic annotation to the penn treebank. *Proceedings of the Human Language Technology Conference (HLT'02)*.

Marcus, M., Marcinkiewicz, M., & Santorini, B. (1993). Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*.

Nakov, P., & Hearst, M. (2008). Solving relational similarity problems using the web as a corpus. *Proceedings of ACL*.

Schulte im Walde, S., Melinger, A., Roth, M., & Weber, A. (2008). An empirical characterisation of response types in german association norms. *To appear, Research on Language and Computation*.

Snow, R., Jurafsky, D., & Ng, A. (2006). Semantic taxonomy induction from heterogenous evidence. *Proceedings of COLING/ACL*.

von Ahn, L., & Dabbish, L. (2004). Labeling images with a computer game. *ACM CHI*.

von Ahn, L., Kedia, M., & Blum, M. (2006). Verbosity: a game for collecting common-sense facts. *Proceedings of the SIGCHI conference on Human Factors in computing systems*.

Seed and Grow: Augmenting Statistically Generated Summary Sentences using Schematic Word Patterns

Stephen Wan[†‡] **Robert Dale**[†] **Mark Dras**[†]
[†]Centre for Language Technology
Department of Computing
Macquarie University
Sydney, NSW 2113
swan,madras,rdale@ics.mq.edu.au

Cécile Paris[‡]
[‡]ICT Centre
CSIRO
Sydney, Australia
Cecile.Paris@csiro.au

Abstract

We examine the problem of content selection in statistical novel sentence generation. Our approach models the processes performed by professional editors when incorporating material from additional sentences to support some initially chosen key summary sentence, a process we refer to as *Sentence Augmentation*. We propose and evaluate a method called "Seed and Grow" for selecting such auxiliary information. Additionally, we argue that this can be performed using schemata, as represented by word-pair co-occurrences, and demonstrate its use in statistical summary sentence generation. Evaluation results are supportive, indicating that a schemata model significantly improves over the baseline.

1 Introduction

In the context of automatic text summarisation, we examine the problem of statistical novel sentence generation, with the aim of moving from the current state-of-the-art of sentence extraction to abstract-like summaries. In particular, we focus on the task of selecting content to include within a generated sentence.

Our approach to novel sentence generation is to model the processes underlying summarisation as performed by professional editors and abstractors. An example of the target output of this kind of generation is presented in Figure 1. In this example, the human authored summary sentence was taken verbatim from the executive summary of a United Nations proposal for the provision of aid addressing a particular humanitarian crisis. Such documents typically exceed a hundred pages.

Human-Authored Summary Sentence:
Repeated [poor seasonal rains]$_1$ [in 2004]$_2$, culminating in [food insecurity]$_3$, indicate [another year]$_4$ of crisis, the scale of which is larger than last year's and is further [exacerbated by diminishing coping assets]$_5$ [in both rural and urban areas]$_6$.

Key Source Sentence:
The consequences of [another year]$_4$ of [poor rains]$_1$ on [food security]$_3$ are severe.

Auxiliary Source Sentence(s):
However in addition to the needs of economic recovery activities for IDPs, [food insecurity]$_3$ [over the majority of 2004]$_2$ [has created great stress]$_5$ on the poorest families in the country, [both within the urban and rural settings]$_6$.

Figure 1: Alignment of a summary sentence to sentences in the full document. Phrases of similar meaning are co-indexed.

To write such summaries, we assume that the human abstractor begins by choosing key sentences from the full document. Then, for each key sentence, a set of auxiliary material is identified. The key sentence is revised incorporating these auxiliary sentences to produce the eventual summary sentence.

To study this phenomenon, a corpus of UN documents was collected and analysed.[1] Each document was divided into two parts comprising its executive summary, and the remainder, referred to here as the *source*. We manually aligned each executive summary sentence with one or more sentences from the source, by choosing a key sentence that provided

[1]This corpus is described in detail in Section 5.1.

evidence for the content of the summary sentence along with additional sentences that provided supporting material.

We refer to the resulting corpus as the UN Consolidated Appeals Process (UN CAP) corpus. It is a collection of sentence alignments, each referred to as an *aligned sentence tuple*, which consists of:

1. A human authored summary sentence from the executive summary;
2. A *key* sentence from the *source*;
3. Zero or more *auxiliary* sentences from the *source*.

The key and any auxiliary sentences are referred to collectively as the *aligned source sentences*.

We argue that some process that combines information from multiple sentences is required if we are to generate summary sentences similar to that portrayed in Figure 1. This is supported by our analysis of the UN CAP corpus. Of the 580 aligned sentence tuples, the majority, 61% of cases, appear to be examples of such a process.

Furthermore, the auxiliary sentences are clearly necessary. We found that only 30% of the open-class words in the summary are found in the key sentence. If one selects all the open-class words from aligned source sentences, recall increases to an upper limit of 45% without yet accounting for stemming. This upper bound is consistent with the upper limit of 50% found by Daumé III and Marcu (2005) which takes into account stemming differences.

This demonstrates that the auxiliary material is a valuable source of content which should be integrated into the summary sentence, allowing an improvement in recall of up to 15% prior to accounting for morphological, synonym and paraphrase differences. Of course, the trick is to improve recall without hurting precision. A naive addition of all words in the aligned source sentences incurs a drop in precision from 30% to 23%. The problem thus is one of selecting the relevant auxiliary content words without introducing unimportant content. We refer to this problem of incorporating material from auxiliary sentences to supplement a key sentence as *Sentence Augmentation*.

In this paper, sentence augmentation is modelled as a noisy channel process and has two facets: content selection and language modelling. This paper focuses on the former, in which the system must rank text segments—in this case, words—for inclusion in the generated sentence. Given a ranked selection of words, a language model would then order them appropriately, as described in work on sentence regeneration (for example, see Soricut and Marcu (2005); Wan et al. (2005)).

Provided with an aligned sentence tuple, the problem lies in effectively selecting words from the auxiliary sentences to bolster those taken from the key sentence. Given that there are on average 2.7 auxiliary sentences per aligned sentence tuple, this additional influx of words poses a considerable challenge.

We begin with the premise that, for documents of a homogeneous type (in this case, the genre is a funding proposal, and the domain is humanitarian aid), it may be possible to identify patterns in the organisation of information in summaries. For example, Figure 2 presents three summary sentences from our corpus that share the same patterned juxtaposition of two concepts *DisplacedPersons* and *HostingCommunities*. Documents may exhibit common patterns since they have a similar goal: namely, to convince donors to give financial support. In the above example, the juxtaposition highlights the fact that those in need are not just those people from the 'epicenter' of the crisis but also those that look after them.

We propose and evaluate a method called "Seed and Grow" for selecting content from auxiliary sentences. That is, we first select the core meaning of the summary, given here by the key sentence, and then we find those pieces of *additional* information that are conventionally juxtaposed with it.

Such patterns are reminiscent of *Schemata*, the organisations of propositional content introduced by McKeown (1985). Schemata typically involve a symbolic representation of each proposition's semantics. However, in our case, a text-to-text generation scenario, we are without such representations and so must find other means to encode these patterns.

To alleviate the situation, we turn to word-pair co-occurrences to approximate schematic patterns. Fig-

Sentence 1:
The increased number of [internally displaced persons]$_1$ and the continued presence of refugees have further strained the scarce natural resources of [host communities]$_2$, stretching their capacity to the limit.

Sentence 2:
100,000 people, a significant portion of the population, remain [displaced]$_1$, burdening the already precarious living conditions of [host families]$_2$ in Dili and the Districts.

Sentence 3:
The current humanitarian situation in Timor-Leste is characterised by: An estimated [100,000 displaced people]$_1$ (10% of the population) living in camps and with [host families]$_2$ in the districts; A total or partial destruction of over 3,000 homes in Dili affecting at least 14,000 IDPs

Figure 2: Examples of the pattern $\langle$*DisplacedPersons*[1], *HostingCommunities*[2]$\rangle$.

ure 2 showed that mentions of the plight of internationally displaced persons are often followed by descriptions of the impact on the host communities that look after them. In this particular example, this is realised lexically in the co-occurrences of the words *displaced* and *host*.

Corpus-based methods inspired by the notion of schemata have been explored in the past by Lapata (2003) and Barzilay and Lee (2004) for ordering sentences extracted in a multi-document summarisation application. However, to our knowledge, using word co-occurrences in this manner to represent schematic knowledge for the purposes of selecting content in a statistically-generated summary sentence has not previously been explored.

This paper seeks to determine whether or not such patterns exist in homogeneous data; and furthermore, whether such patterns can be used to better select words from auxiliary sentences. In particular, we propose the "Seed and Grow" approach for this task. The results show that even simple modelling approaches are able to model this schematic information.

In the remainder of this paper, we contrast our approach to related text-to-text research in Section 2. The Content Selection model is presented in Section 3. Section 4 describes how a binary classification model is used in a statistical text generation system. Section 5 describes our evaluation of the model for a summary generation task. We conclude, in Section 6, that domain-specific schematic patterns can be acquired and applied to content selection for statistical sentence generation.

2 Related Work

2.1 Content Selection in Text-to-Text Systems

Statistical text-to-text summarisation applications have borrowed much from the related field of statistical machine translation. In one of the first works to present summarisation as a noisy channel approach, Witbrock and Mittal (1999) presented a conditional model for learning the suitability of words from a news article for inclusion in headlines, or 'ultra-summaries'. Inspired by this approach, and with the intention of designing a robust statistical generation system, our work is also based on the noisy channel model. Into this, we incorporate our content selection model, which includes Witbrock and Mittal's model supplemented with schema-based information.

Roughly, text-to-text transformations fall into three categories: those in which information is *compressed*, *conserved*, and *augmented*. We use these distinctions to organise this overview of the literature.

In *Sentence Compression* work, a single sentence undergoes pruning to shorten its length. Previous approaches have focused on statistical syntactic transformations (Knight and Marcu, 2002). For content selection, discourse-level considerations were proposed by Daumé III and Marcu (2002), who explored the use of Rhetorical Structure Theory (Mann and Thompson, 1988). More recently, Clarke and Lapata (2007) use Centering Theory (Grosz et al., 1995) and Lexical Chains (Morris and Hirst, 1991) to identify which information to prune. Our work is similar in incorporating discourse-level phenomena for content selection. However, we look at schema-like information as opposed to chains of references and focus on the sentence augmentation task.

The work of Barzilay and McKeown (2005) on *Sentence Fusion* introduced the problem of converting multiple sentences into a single summary sentence. Each sentence set ideally tightly clusters around a single news event. Thus, there is one general proposition to be realised in the summary sentence, identified by finding the common elements in the input sentences. We see this as an example of *conservation*. In our work, this general proposition is equivalent to the core information for the summary sentence *before* the incorporation of supplementary material.

In contrast to both *compression* and *conservation* work, we focus on *augmenting* the information in a key sentence. The closest work is that of Jing and McKeown (1999) and Daumé III and Marcu (2005), in which multiple sentences are processed, with fragments within them being recycled to generate the novel generated text.

In both works, recyclable fragments are identified by automatic means. Jing and McKeown (1999) use models that are based on "copy-and-paste" operations learnt from the behaviour of human abstractors as found in a corpus. Daumé III and Marcu (2005) propose a model that encodes how likely it is that different sized spans of text are skipped to reach words and phrases to recycle.

While similar in task, our models differ substantially in the nature of the phenomenon modelled. In this work, we focus on content-based considerations that model which words can be combined to build up a new sentence.

2.2 Schemata and Text Generation

There exists related work from Natural Language Generation (NLG) in finding material to build up sentences. As mentioned above, our content selection model is inspired by work on schemata from NLG (McKeown, 1985). Barzilay and Lee (2004) showed that it is possible to obtain schema-like knowledge automatically from a corpus for the purposes of *extracting* sentences and ordering them. However, their work represents patterns at the sentence level, and is thus not directly comparable to our work, given our focus on sentence *generation*.

In our system, what is required is a means to rank words for use in generation. Thus, we focus on commonly occurring word co-occurrences, with the aim of encoding conventions in the texts we are trying to generate. In this respect, this is similar to work by Lapata (2003), who builds a conditional model of words across adjacent sentences, focusing on words in particular semantic roles. Like Barzilay and Lee (2004), this model was used to order extracted sentences in summaries. In contrast, our work focuses on word patterns found within a summary sentence, not between sentences. Additionally, our tasks differ as we examine the statistical sentence generation instead of sentence ordering.

3 Linguistic Intuitions behind Word Selection

The "Seed and Grow" approach proposed in this paper divides the word-level content selection problem into two underlying subproblems. We address these with two separate models, called the *salience* and *schematic* models. The salience model chooses the key content for the summary sentence while the schematic model attempts to identify what else is typically mentioned given those salient pieces of information.

3.1 A Salience Model: Learning "Buzzwords"

There are a variety of methods for determining the salient information in a text, and these underpin most work in automatic text summarisation. As an example of a salience model trained on corpus data, Witbrock and Mittal (1999) introduced a method for scoring summary words for inclusion within news headlines. In their model, headlines were treated as 'ultra-summaries'. Their model learns which words are typically used in headlines and encodes, at least to some degree, which words are attention grabbing.

In the domain of funding proposals, key words that grab attention may amount to domain-specific buzzwords. Intuitively, a reader, perhaps someone in charge of allocating donations, tends to look for certain types of key information matching donation criteria, and so human abstract authors will target their summaries for this purpose.

We thus adapt the Witbrock and Mittal (1999) model to identify such domain specific buzzwords (BWM, for 'buzzword model'). For an aligned sentence tuple, the probability that a word is selected based on the salience of a word with respect to the domain is defined as:

$$\text{prob}_{bwm}(\text{select} = 1|w) = \frac{|\text{summary}_w|}{|\text{source}_w|} \qquad (1)$$

where summary$_w$ is the set of aligned sentence tuples that contain the word *w* in the summary sentence *and* in the source sentences. The denominator, source$_w$, is the set of aligned sentence tuples that have the word *w* in either the key or an auxiliary sentence.

As is implicit in this equation, we could just use this buzzword model to select content not only from the key sentence, but from the auxiliary sentences as well. While it is intended ultimately to find the key content of the summary, it can also serve as an alternative baseline for auxiliary content selection to compare against the "Seed and Grow" model.

3.2 A Schema Model: Approximation via Word co-Occurrences

To restate the problem at hand: the task is one of finding elements of secondary importance that schematically elaborate on the key information. We do this by examining sample summary sentences for conventional juxtapositions of concepts. As mentioned in Section 1, schemata are approximated here with patterns of word-pair co-occurrences. Using a corpus of human-authored summaries in the domain of our application, it is thus possible to learn what those common combinations of words are.

Roughly, the process is as follows. To begin with, a *seed set* of words is chosen. The purpose of the seed set is to represent the core proposition of the summary sentence.

In this work, this core proposition is given by the key sentence and so the non-stopwords belonging to it are used to populate the seed set. In the "Seed and Grow" approach, we check to see which words from auxiliary sentences pair well with words in the seed set.

3.2.1 Collecting Word-level Patterns

Each training case in the corpus contains a single human-authored summary sentence that can be used to learn which pairs of words conventionally occur in a summary. For each summary sentence, stopwords are removed. Then, each pairing of words in the sentence is used to update a pair-wise word co-occurrence frequency table. When looking up and storing a frequency, the order of words is ignored.

3.2.2 Scoring Word-Pair Co-occurrence Strength

For any two words, w_1 from the seed set and w_2 from an auxiliary sentence, the word-pair co-occurrence probability is defined as follows:

$$\begin{aligned} & \text{prob}_{co\text{-}oc}(w_1, w_2) \\ & = \frac{\text{freq}(w_1, w_2)}{\text{freq}(w_1) + \text{freq}(w_2) - \text{freq}(w_1, w_2)} \end{aligned} \quad (2)$$

where $freq(w_1, w_2)$ is a lookup in the word-pair co-occurrence frequency table. This table stores co-occurrence word pairs occurring in the summary sentence.

3.2.3 Combining a Set of Co-occurrence Scores

Each auxiliary word now has a series of scores, one for each comparison with a seed word. To rank each auxiliary word, these need to be combined into a single score for sorting.

When combining the set of co-occurrence scores, one might want to account for the fact that each pairing of a seed word with an auxiliary word might not contribute equally to the overall selection of that auxiliary word. Intuitively, a word in the seed set, derived from the key sentence, may only make a minor contribution to the core meaning of the summary sentence. For example, words that are part of an adjunct phrase in the key sentence might not be good candidates to elaborate upon. Thus, one might want to weight these seed words lower, to reduce their influence on triggering schematically associated words.

To allow for this, a seed weight vector is maintained, storing a weight per seed word. Different weighting schemes are possible. For example, a scheme might indicate the salience of a word. In addition to the buzzword model (BWM) described earlier, one might employ a standard vector space approach (Salton and McGill, 1983) from Information Retrieval, which uses term frequency scores weighted with an inverse document frequency factor, or *tf-idf*. We also implement the case in which all seed words are treated equally using binary weights, where 1 indicates the presence of a seed word, and 0 indicates its absence. In the evaluations described in Section 5, we refer to these three seed weighting schemes as *bwm* and *tf-idf*, and *binary* respectively.

To find the probability of selecting an auxiliary word using the schematic word-pair co-occurrence model (WCM), an averaged probability is found by normalising the sum of the weighted probabilities, where weights are provided by one of the three schemes above:

$$\text{prob}_{wcm}(w_i) = \frac{1}{Z} \times \sum_{k=0}^{|\text{seed}|} \text{weights}_k \times \text{prob}_{co\text{-}oc}(w_i, w_k) \quad (3)$$

where *seed* is the set of seed words and w_k is the k^{th} word in that set. The vector, *weights*, stores the seed weights. The normalisation factor for the weighted average, Z, is the number of auxiliary words.

Finally, since the WCM model only serves to select words from the *auxiliary* sentences, words from the key sentence must be given scores as well. For these words, the scoring is as follows:

$$\text{prob}_{wcm}(w) = \frac{1}{Z}\left(\frac{1}{|\text{seed}|} + \text{prob}_{wcm}(w)\right) \quad (4)$$

where Z is a normalisation across the set of seed words.

4 Combining Buzzwords and Word-Pair Co-Occurrence Models for Generation

As mentioned above, the noisy channel approach is used for producing the augmented sentence. Although the focus of this paper is on Content Selection, an overview of the end-to-end generation process is presented for completeness.

Sentence augmentation is essentially a text-to-text process: A key sentence and auxiliary material are transformed into a single summary sentence. Following Witbrock and Mittal (1999), the task is to search for the string of words that maximises the probability $prob(summary|source)$. Standardly reformulating this probability using Bayes' rule results in the following:

$$prob_{cm}(source|summary) \times prob_{lm}(summary) \quad (5)$$

In this paper, we are concerned with the first factor, $prob_{cm}(source|summary)$, referred to as the channel model (CM), which combines both the buzzword (BWM) and word-pair co-occurrence (WCM) models. An examination of differences between the two approaches revealed only a 20% word overlap on the Jaccard metric.

In order to combine multiple models, we intend to use machine learning approaches to combine the information in each model in a similar manner to Berger et al. (1996). We are currently exploring the use of logistic regression methods to learn a function that would treat, as features, the probabilities defined by the salience and schematic content selection models. Although generation is possible using each content selection model in isolation, evaluations of the combined model are on-going and are not presented in this paper.

5 Evaluation

In this evaluation, the task is to select n words from the aligned source sentences for inclusion in a summary. As a gold-standard for comparison, we simply examine what words were actually chosen in the summary sentence of the aligned sentence tuple. We are specifically interested in open-class words, and so a stopword list of closed-class words is used to filter the sentences in each test case.

We evaluate against the set of open-class words in the human-authored summary sentence using recall and precision metrics. Recall is the size of the intersection of the selected and gold-standard sets, normalised by the length of the gold-standard sentence (in words). This recall metric is similar to the ROUGE-1 metric, the unigram version of the ROUGE metric (Lin and Hovy, 2003) used in the Document Understanding Conferences[2] (DUC). Precision is the size of the intersection normalised by the number of words selected. We also report the F-measure, which is the harmonic mean of the recall and precision scores.

Recall, precision and F-measure are measured at various values of n ranging from 1 to the number of open-class words in the gold-standard summary sentence for a particular test case. For the purposes of evaluation, differences in tokens due to morphology were explored crudely via the use of Porter's stemming algorithm. However, the results from stemming are not that different from exact token matches when examining performance on the entire data set

[2]http://duc.nist.gov

Number of training cases	530
Average words in summary sentence	27.0
Average stopwords in summary sentence	10.3
Average number of auxiliary sentences	2.75
Word count: summary sentences	4630
Word count: source sentences	21356
Word type count in corpus	3800

Table 1: Statistics for the UN CAP training set

and so, for simplicity, these are omitted in this discussion.

5.1 The Data

The corpus is made up of a number of humanitarian aid proposals called Consolidated Appeals Process (UN CAP) documents, which are archived at the United Nations website.[3] 135 documents from the period 2002 to 2007 were downloaded by the authors. A preprocessing stage extracted text from the PDF files and segmented the documents into executive summary and source sections. These were then automatically segmented further into sentences.

Executive summary sentences were manually aligned by the authors to source key and auxiliary sentences, producing a corpus of 580 aligned sentence tuples referred to here as the UN CAP corpus. Of these, 230 tuples were paraphrase cases (i.e. without aligned auxiliary sentences). The remaining 550 cases were instances of sentence augmentation (with at least one auxiliary sentence).

Of the 580 cases, 50 cases were set aside for testing. The remaining 530 cases were used for training. Statistics for the training portion of the sentence augmentation set are provided in Table 1.

In this paper, aligned sentence tuples are obtained via manual annotation. Automatic construction of these sentence-level alignments is possible and has been explored by Jing and McKeown (1999). We also envisage using tools for scoring sentence similarity (for example, see Hatzivassiloglou et al. (2001)) for automatically constructing them; this is the focus of work by Wan and Paris (2008).

[3]http://ochaonline3.un.org/humanitarianappeal/index.htm

5.2 The Baselines

Three baselines were used in this work: the *random*, *tf-idf* and *position* baselines. A *random* word selector shows what performance might be achieved in the absence of any linguistic knowledge.

We also sorted all words in the aligned source sentences by their weighted *tf-idf* scores. This baseline selects words in order until the desired word limit is reached. This baseline is referred to as the *tf-idf* baseline.

Finally, we selected words based on their sentence order, choosing first those words from the key sentence. When these are exhausted, auxiliary sentences are sorted by their sentence positions in the original document. Words from the first auxiliary sentence are then chosen. This continues until either the desired number of words have been chosen, or no words remain. This baseline is known as the *position* baseline.

5.3 Content Selection Results

We compare the three baselines to the two models presented in Section 3. These are the buzzword salience model (BWM) and the schematic word-pair co-occurrence model (WCM).

We begin by presenting recall, precision and F-measure graphs when selecting from the aligned source sentences, comprising the key and auxiliary sentences. Figure 3 shows the results for the two models against the three baselines. The two models, the positional, and the *tf-idf* baselines perform better than the random baseline, as measured by a two-tailed Wilcoxon Matched Pairs Signed Ranks test ($\alpha = 0.05$).

The WCM consistently out-performs the BWM on all metrics, and the differences are statistically significant. In fact, the BWM also generally performs worse than the position and *tf-idf* baselines. WCM and the position baseline both significantly outperform the *tf-idf* baseline on all metrics for longer sentence lengths.

That the position baseline and WCM should perform similarly is not really surprising since, in effect, the position baseline first chooses words from the key sentence and then selects auxiliary words. The difference essentially lies in how the auxiliary words are chosen.

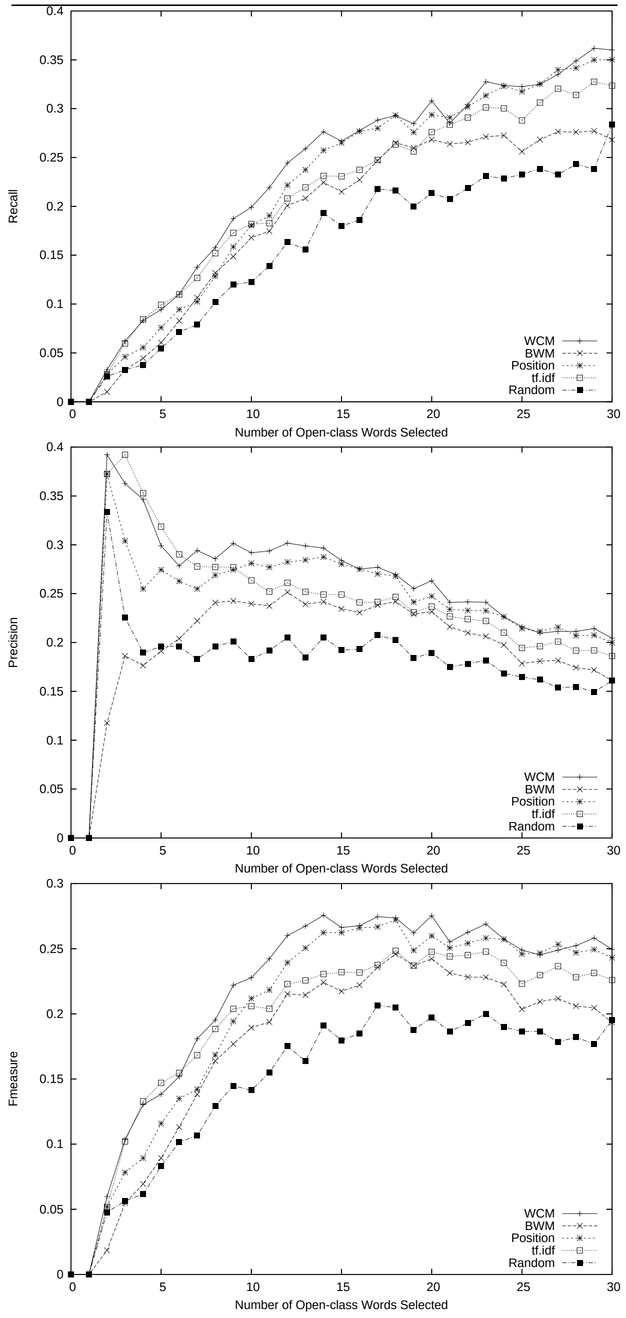


Figure 3: Recall, Precision and F-measure performance for open-class words from the entire input set (key and auxiliary). Models presented are the Buzzword Model (BWM), the Word-Pair Co-occurrence Model (WCM) and position, *tf-idf* and random baselines.

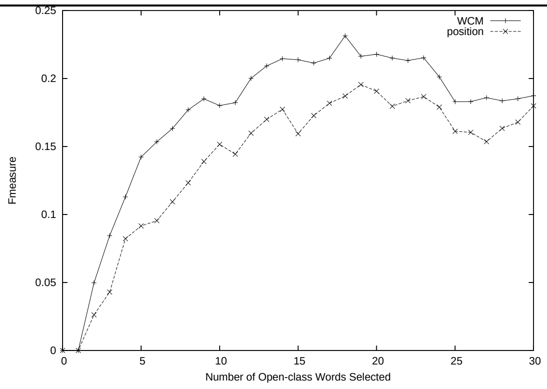


Figure 4: F-measure scores for content selection on just the auxiliary sentences. Models presented are the Word-Pair Co-occurrence model (WCM) and the position baseline.

The results of Figure 3 weakly support the hypothesis that using schematic word-pair co-occurrences helps improve performance over models without discourse-related features. The graphs show that WCM edges above the position baseline when the number of selected open-class words ranges from 10 to 15. Note that the average number of open-class words in a human authored summary sentence is 16. The only significant difference found was in the F-measure and precision scores for 19 selected open-class words. Nevertheless, a general trend can be observed in which WCM performs better than the position baseline.

Ultimately, however, what we want to do is select auxiliary content to supplement the key sentence. To examine the effect of two best performing approaches, WCM and the position baseline, on this task, were both modified so that the key sentence words were explicitly given a zero probability. Thus, the recall, precision and F-measure scores obtained are based solely on the ability of either to select auxiliary words. The F-measure scores are presented Figure 4. WCM consistently outperforms the position baseline for the selection of auxiliary words. Differences are significant for 6 or more selected open-class words.

The results show that even when considering only exact token matches, we can improve on the recall of open-class words, and do so without penalty in precision. Our working hypothesis is that such gains are possible because the corpus has a homogeneous quality and key patterns are sufficiently repeated even when the overall data set is of the order of hundreds of cases. The benefit of using a model encoding some schematic information is further shown by the performance of WCM over the position baseline when selecting words from auxiliary sentences.

This is an interesting finding given that domain independent methods are increasingly used on domain-specific corpora such as financial and biomedical texts, for which we may have access to only a limited amount of data. We anticipate that as we introduce methods to account for paraphrase and synonym differences, performance might rise further still.

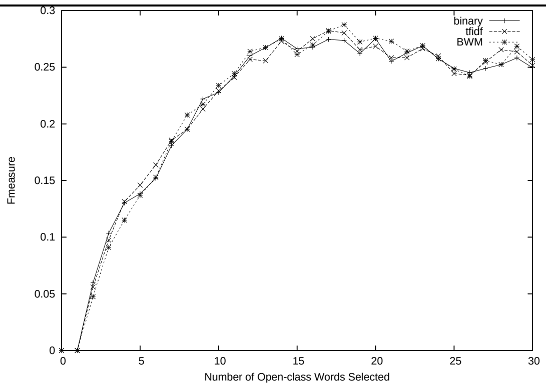


Figure 5: F-measure performance for open-class words from the entire input set (key and auxiliary). Models presented are variants of the Word-Pair Co-occurrence Model (WCM) that differ in the seed weighting schemes.

5.4 Testing Seed Weighting Schemes

We can also weight seed words in the "Seed and Grow" approach in a variety of ways. To test whether weighting schemes have any effect on content selection performance, we examined the use of three schemes. We were particularly interested in those schemes that indicate the contribution of a seed word to the core meaning of a sentence. These are the *binary*, *tf-idf* and *buzzword* weighting schemes described in Section 3. We present the F-measure graph for these three variants of the schematic word-pair co-occurrence model (WCM) in Figure 5.

The graphs show that there is no discernible difference between the seed weighting schemes. No scheme significantly outperforms another. Thus, we conclude that the choice of these particular seed weighting schemes has no effect on performance. In future work, we intend to examine whether weighting schemes encoding syntactic information might fare better, since such information might more accurately represent the contribution of a substring to the main clause of the sentence.

6 Conclusions and Future Work

In this paper, we argued a case for *sentence augmentation*, a component that facilitates abstract-like text summarisation. We showed that such a process can account for summary sentences as authored by professional editors. We proposed the use of schemata, as approximated with a word-pair co-occurrence model, and advocated a new schema-based "Seed and Grow" content selection model used for statistical sentence generation.

We also showed that domain-specific patterns, schematic word-pair co-occurrences in this case, can be acquired from a limited amount of data as indicated by modest performance gains for content selection using schemata information. We postulate that this is particularly true when dealing with homogeneous data.

In future work, we intend to explore other string matches corresponding to variations due to paraphrases and synonymy. We would also like to study the effects of corpus size when learning schematic patterns. Finally, we are currently investigating the use of machine learning methods to combine the best of the Salience and Schemata models in order to provide a single model for use in decoding.

7 Acknowledgments

We would like to thank the reviewers for their insightful comments. This work was funded by the CSIRO ICT Centre and Centre for Language Technology at Macquarie University.

References

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 113–120, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Regina Barzilay and Kathleen R. McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

Adam L. Berger, Stephen Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

James Clarke and Mirella Lapata. 2007. Modelling compression with discourse constraints. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1–11.

Hal Daumé III and Daniel Marcu. 2002. A noisy-channel model for document compression. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL – 2002)*, pages 449 – 456, Philadelphia, PA, July 6 – 12.

Hal Daumé III and Daniel Marcu. 2005. Induction of word and phrase alignments for automatic document summarization. *Computational Linguistics*, 31(4):505–530, December.

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.

V. Hatzivassiloglou, J. Klavans, M. Holcombe, R. Barzilay, M. Kan, and K. McKeown. 2001. Simfinder: A flexible clustering tool for summarization. pages 41–49. Association for Computational Linguistics.

Hongyan Jing and Kathleen McKeown. 1999. The decomposition of human-written summary sentences. In *Research and Development in Information Retrieval*, pages 129–136.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: a probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

Mirella Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 545–552, Sapporo, Japan.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 71–78, Morristown, NJ, USA. Association for Computational Linguistics.

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

Kathleen R McKeown. 1985. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cambridge University Press.

Jane Morris and Graeme Hirst. 1991. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17(1):21–48.

G. Salton and M. J. McGill. 1983. *Introduction to modern information retrieval*. McGraw-Hill, New York.

Radu Soricut and Daniel Marcu. 2005. Towards developing generation algorithms for text-to-text applications. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 66–74, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Stephen Wan and Cécile Paris. 2008. In-browser summarisation: Generating elaborative summaries biased towards the reading context. In *Proceedings of ACL-08: HLT, Short Papers*, pages 129–132, Columbus, Ohio, June. Association for Computational Linguistics.

Stephen Wan, Robert Dale Mark Dras, and Cécile Paris. 2005. Towards statistical paraphrase generation: preliminary evaluations of grammaticality. In *Proceedings of The 3rd International Workshop on Paraphrasing (IWP2005)*, pages 88–95, Jeju Island, South Korea.

Michael J. Witbrock and Vibhu O. Mittal. 1999. Ultra-summarization (poster abstract): a statistical approach to generating highly condensed non-extractive summaries. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 315–316, New York, NY, USA. ACM Press.

Using Bilingual Knowledge and Ensemble Techniques for Unsupervised Chinese Sentiment Analysis

Xiaojun Wan
Institute of Compute Science and Technology
Peking University
Beijing 100871, China
wanxiaojun@icst.pku.edu.cn

Abstract

It is a challenging task to identify sentiment polarity of Chinese reviews because the resources for Chinese sentiment analysis are limited. Instead of leveraging only monolingual Chinese knowledge, this study proposes a novel approach to leverage reliable English resources to improve Chinese sentiment analysis. Rather than simply projecting English resources onto Chinese resources, our approach first translates Chinese reviews into English reviews by machine translation services, and then identifies the sentiment polarity of English reviews by directly leveraging English resources. Furthermore, our approach performs sentiment analysis for both Chinese reviews and English reviews, and then uses ensemble methods to combine the individual analysis results. Experimental results on a dataset of 886 Chinese product reviews demonstrate the effectiveness of the proposed approach. The individual analysis of the translated English reviews outperforms the individual analysis of the original Chinese reviews, and the combination of the individual analysis results further improves the performance.

1 Introduction

In recent years, sentiment analysis (including subjective/objective analysis, polarity identification, opinion extraction, etc.) has drawn much attention in the NLP field. In this study, the objective of sentiment analysis is to annotate a given text for polarity orientation (positive/negative). Polarity orientation identification has many useful applications, including opinion summarization (Ku et al., 2006) and sentiment retrieval (Eguchi and Lavrenko, 2006).

To date, most of the research focuses on English and a variety of reliable English resources for sentiment analysis are available, including polarity lexicon, contextual valence shifters, etc. However, the resources for other languages are limited. In particular, few reliable resources are available for Chinese sentiment analysis[1] and it is not a trivial task to manually label reliable Chinese sentiment resources.

Instead of using only the limited Chinese knowledge, this study aims to improve Chinese sentiment analysis by making full use of bilingual knowledge in an unsupervised way, including both Chinese resources and English resources. Generally speaking, there are two unsupervised scenarios for "borrowing" English resources for sentiment analysis in other languages: one is to generate resources in a new language by leveraging on the resources available in English via cross-lingual projections, and then perform sentiment analysis in the English language based on the generated resources, which has been investigated by Mihalcea et al. (2007); the other is to translate the texts in a new language into English texts, and then perform sentiment analysis in the English language, which has not yet been investigated.

In this study, we first translate Chinese reviews into English reviews by using machine translation services, and then identify the sentiment polarity of English reviews by directly leveraging English resources. Furthermore, ensemble methods are employed to combine the individual analysis results in each language (i.e. Chinese and English) in order to obtain improved results. Given machine translation services between the selected target language and English, the proposed approach can be applied to any other languages as well.

Experiments have been performed on a dataset of 886 Chinese product reviews. Two commercial

[1] This study focuses on Simplified Chinese.

machine translation services (i.e. Google Translate and Yahoo Babel Fish) and a baseline dictionary-based system are used for translating Chinese reviews into English reviews. Experimental results show that the analysis of English reviews translated by the commercial translation services outperforms the analysis of original Chinese reviews. Moreover, the analysis performance can be further improved by combining the individual analysis results in different languages. The results also demonstrate that our proposed approach is more effective than the approach that leverages generated Chinese resources.

The rest of this paper is organized as follows: Section 2 introduces related work. The proposed approach is described in detail in Section 3. Section 4 shows the experimental results. Lastly we conclude this paper in Section 5.

2 Related Work

Polarity identification can be performed on word level, sentence level or document level. Related work for word-level polarity identification includes (Hatzivassiloglou and McKeown, 1997; Kim and Hovy. 2004; Takamura et al., 2005; Yao et al. 2006; Kaji and Kitsuregawa, 2007), and related work for sentence-level polarity identification includes (Yu and Hatzivassiloglou, 2003; Kim and Hovy. 2004) Word-level or sentence-level sentiment analysis is not the focus of this paper.

Generally speaking, document-level polarity identification methods can be categorized into unsupervised and supervised.

Unsupervised methods involve deriving a sentiment metric for text without training corpus. Turney (2002) predicates the sentiment orientation of a review by the average semantic orientation of the phrases in the review that contain adjectives or adverbs, which is denoted as the semantic oriented method. Kim and Hovy (2004) build three models to assign a sentiment category to a given sentence by combining the individual sentiments of sentiment-bearing words. Hiroshi et al. (2004) use the technique of deep language analysis for machine translation to extract sentiment units in text documents. Kennedy and Inkpen (2006) determine the sentiment of a customer review by counting positive and negative terms and taking into account contextual valence shifters, such as negations and intensifiers. Devitt and Ahmad (2007) explore a computable metric of positive or negative polarity in financial news text.

Supervised methods consider the sentiment analysis task as a classification task and use labeled corpus to train the classifier. Since the work of Pang et al. (2002), various classification models and linguistic features have been proposed to improve the classification performance (Pang and Lee, 2004; Mullen and Collier, 2004; Wilson et al., 2005a; Read, 2005). Most recently, McDonald et al. (2007) investigate a structured model for jointly classifying the sentiment of text at varying levels of granularity. Blitzer et al. (2007) investigate domain adaptation for sentiment classifiers, focusing on online reviews for different types of products. Andreevskaia and Bergler (2008) present a new system consisting of the ensemble of a corpus-based classifier and a lexicon-based classifier with precision-based vote weighting.

Research work focusing on Chinese sentiment analysis includes (Tsou et al., 2005; Ye et al., 2006; Li and Sun, 2007; Wang et al., 2007). Such work represents heuristic extensions of the unsupervised or supervised methods for English sentiment analysis.

To date, the most closely related work is Mihalcea et al. (2007), which explores cross-lingual projections to generate subjectivity analysis resources in Romanian by leveraging on the tools and resources available in English. They have investigated two approaches: a lexicon-based approach based on Romanian subjectivity lexicon translated from English lexicon, and a corpus-based approach based on Romanian subjectivity-annotated corpora obtained via cross-lingual projections. In this study, we focus on unsupervised sentiment polarity identification and we only investigate the lexicon-based approach in the experiments.

Other related work includes subjective/objective analysis (Hatzivassiloglon and Wiebe, 2000; Riloff and Wiebe, 2003) and opinion mining and summarization (Liu et al., 2005; Popescu and Etzioni. 2005; Choi et al., 2006; Ku et al., 2006; Titov and McDonald, 2008).

3 The Proposed Approach

3.1 Overview

The motivation of our approach is to make full use of bilingual knowledge to improve sentiment analysis in a target language, where the resources for sentiment analysis are limited or unreliable. This study focuses on unsupervised polarity identification of Chinese product reviews by using both the rich English knowledge and the limited Chinese knowledge.

The framework of our approach is illustrated in Figure 1. A Chinese review is translated into the corresponding English review using machine translation services, and then the Chinese review and the English review are analyzed based on Chinese resources and English resources, respectively. The analysis results are then combined to obtain more accurate results under the assumption that the individual sentiment analysis can complement each other. Note that in the framework, different machine translation services can be used to obtain different English reviews, and the analysis of English reviews translated by a specific machine translation service is conducted separately. For simplicity, we consider the English reviews translated by different machine translation services as reviews in different languages, despite the fact that in essence, they are still in English.

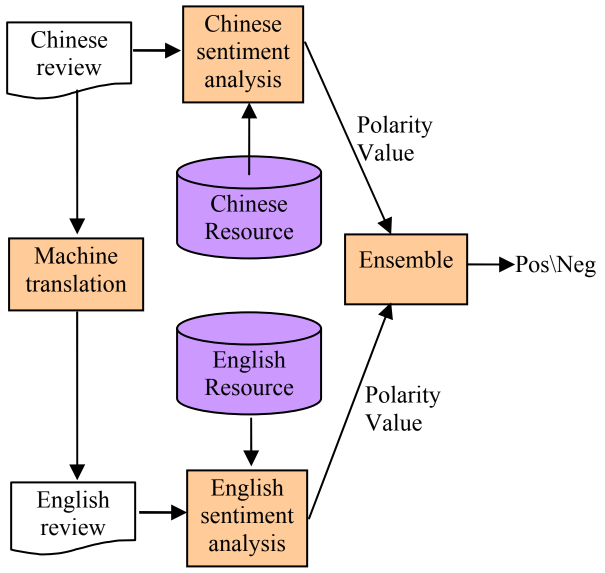


Figure 1. Framework of our approach

Formally, give a review rev^0 in the target language (i.e. Chinese), the corresponding review rev_i in the ith language is obtained by using a translation function: $rev^i = f^i_{Trans}(rev^0)$, where $1 \le i \le p$ and p is the total number of machine translation services. For each review rev^k in the kth language ($0 \le k \le p$), we employ the semantic oriented approach to assign a semantic orientation value $f^k_{SO}(rev^k)$ to the review, and the polarity orientation of the review can be simply predicated based on the value by using a threshold. Given a set of semantic orientation values $F_{SO}=\{f^k_{SO}(rev^k) \mid 0 \le k \le p\}$, the ensemble methods aim to derive a new semantic orientation value $f^{Ensemble}_{SO}(rev^0)$ based on the values in F_{SO}, which can be used to better classify the review as positive or negative.

The steps of review translation, individual semantic orientation value computation and ensemble combination are described in details in the next sections, respectively.

3.2 Review Translation

Translation of a Chinese review into an English review is the first step of the proposed approach. Manual translation is time-consuming and labor-intensive, and it is not feasible to manually translate a large amount of Chinese product reviews in real applications. Fortunately, machine translation techniques have been well developed in the NLP field, though the translation performance is far from satisfactory. A few commercial machine translation services can be publicly accessed. In this study, the following two commercial machine translation services and one baseline system are used to translate Chinese reviews into English reviews.

Google Translate [2] (***GoogleTrans***): Google Translate is one of the state-of-the-art commercial machine translation systems used today. Google Translate applies statistical learning techniques to build a translation model based on both monolingual text in the target language and aligned text consisting of examples of human translations between the languages.

Yahoo Babel Fish [3] (***YahooTrans***): Different from Google Translate, Yaho Babel Fish uses SYSTRAN's rule-based translation engine. SYSTRAN was one of the earliest developers of machine translation software. SYSTRAN applies complex sets of specific rules defined by linguists to analyze and then transfer the grammatical structure of the source language into the target language.

Baseline Translate (*DictTrans*): We simply develop a translation method based only on one-to-one term translation in a large Chinese-to-English

[2] http://translate.google.com/translate_t
[3] http://babelfish.yahoo.com/translate_txt

dictionary. Each term in a Chinese review is translated by the first corresponding term in the Chinese-to-English dictionary, without any other processing steps. In this study, we use the LDC_CE_DIC2.0[4] constructed by LDC as the dictionary for translation, which contains 128366 Chinese terms and their corresponding English terms.

The Chinese-to-English translation performances of the two commercial systems are deemed much better than the weak baseline system. Google Translate has achieved very good results on the Chinese-to-English translation tracks of NIST open machine translation test (MT)[5] and it ranks the first on most tracks. In the Chinese-to-English task of MT2005, the BLEU-4 score of Google Translate is 0.3531, and the BLEU-4 score of SYSTRAN is 0.1471. We can deduce that Google Translate is better than Yahoo Babel Fish, without considering the recent improvements of the two systems.

Here are two running example of Chinese reviews and the translated English reviews (*HumanTrans* refers to human translation):

***Positive Example:** 优点很多,外形也很好。*

HumanTrans: Many advantages and very good shape.

GoogleTrans: Many advantages, the shape is also very good.

YahooTrans: Merit very many, the contour very is also good.

DictTrans: merit very many figure also very good

***Negative example:** 内存太小不支持红外。*

HumanTrans: The memory is too small to support IR.

GoogleTrans: Memory is too small not to support IR.

YahooTrans:The memory too is small does not support infrared.

DictTrans: memory highest small negative not to be in favor of ir.

3.3 Individual Semantic Orientation Value Computation

For any specific language, we employ the semantic orientated approach (Kennedy and Inkpen, 2006) to compute the semantic orientation value of a review. The unsupervised approach is quite straightforward and it makes use of the following sentiment lexicons: **positive Lexicon (*Positive_Dic*)** including terms expressing positive polarity, **Negative Lexicon (*Negative_Dic*)** including terms expressing negative polarity, **Negation Lexicon (*Negation_Dic*)** including terms that are used to reverse the semantic polarity of a particular term, and **Intensifier Lexicon (*Intensifier_Dic*)** including terms that are used to change the degree to which a term is positive or negative. In this study, we conduct our experiments within two languages, and we collect and use the following popular and available Chinese and English sentiment lexicons[6], without any further filtering and labeling:

1) Chinese lexicons

***Positive_Dic^{cn}*:** 3730 Chinese positive terms were collected from the Chinese Vocabulary for Sentiment Analysis (VSA)[7] released by HOWNET.

***Negative_Dic^{cn}*:** 3116 Chinese negative terms were collected from Chinese Vocabulary for Sentiment Analysis (VSA) released by HOWNET.

***Negation_Dic^{cn}*:** 13 negation terms were collected from related papers.

***Intensifier_Dic^{cn}*:** 148 intensifier terms were collected from Chinese Vocabulary for Sentiment Analysis (VSA) released by HOWNET.

2) English lexicons

***Positive_Dic^{en}*:** 2718 English positive terms were collected from the feature file *subjclueslen1-HLTEMNLP05.tff* [8] containing the subjectivity clues used in the work (Wilson et al., 2005a; Wilson et al., 2005b). The clues in this file were collected from a number of sources. Some were culled from manually developed resources, e.g. *general inquirer*[9] (Stone et al., 1966). Others were identified automatically using both annotated and unannotated data. A majority of the clues were collected as part of work reported in Riloff and Wiebe (2003).

***Negative_Dic^{en}*:** 4910 English negative terms were collected from the same file described above.

***Negation_Dic^{en}*:** 88 negation terms were collected from the feature file *valenceshifters.tff* used in the work (Wilson et al., 2005a; Wilson et al., 2005b).

***Intensifier_Dic^{en}*:** 244 intensifier terms were collected from the feature file *intensifiers2.tff* used in the work (Wilson et al., 2005a; Wilson et al., 2005b).

[4] http://projects.ldc.upenn.edu/Chinese/LDC_ch.htm

[5] http://www.nist.gov/speech/tests/mt/

[6] In this study, we focus on using a few popular resources in both Chinese and English for comparative study, instead of trying to collect and use all available resources.

[7] http://www.keenage.com/html/e_index.html

[8] http://www.cs.pitt.edu/mpqa/

[9] http://www.wjh.harvard.edu/~inquirer/homecat.htm

The semantic orientation value $f^k_{SO}(rev^k)$ for rev^k is computed by summing the polarity values of all words in the review, making use of both the word polarity defined in the positive and negative lexicons and the contextual valence shifters defined in the negation and intensifier lexicons. The algorithm is illustrated in Figure 2.

```
Input: a review rev^k in the kth language. Four lexicons in the kth language: Positive_Dic^k, Negative_Dic^k, Negation_Dic^k, Intensifier_Dic^k, which are either Chinese or English lexicons;
Output: Polarity Value $f^k_{SO}(rev^k)$;
Algorithm Compute_SO:
1. Tokenize review $rev_k$ into sentence set S and each sentence $s \in S$ is tokenized into word set $W_s$;
2. For any word w in a sentence $s \in S$, compute its SO value SO(w) as follows:
    1) if $w \in Positive\_Dic^k$, SO(w)=PosValue;
    2) If $w \in Negative\_Dic^k$, SO(w)=NegValue;
    3) Otherwise, SO(w)=0;
    4) Within the window of q words previous to w, if there is a term $w' \in Negation\_Dic^k$, SO(w)= –SO(w);
    5) Within the window of q words previous to w, if there is a term $w' \in Intensifier\_Dic^k$, SO(w) =ρ×SO(w);
3. $f^k_{SO}(rev^k) = \sum_{s \in S} \sum_{w \in W_s} SO(w)$;
```

Figure 2. The algorithm for semantic orientation value computation

In the above algorithm, *PosValue* and *NegValue* are the polarity values for positive words and negative words respectively. We empirically set *PosValue*=1 and *NegValue*= –2 because negative words usually contribute more to the overall semantic orientation of the review than positive words, according to our empirical analysis. ρ>1 aims to intensify the polarity value and we simply set ρ=2. q is the parameter controlling the window size within which the negation terms and intensifier terms have influence on the polarity words and here q is set to 2 words. Note that the above parameters are tuned only for Chinese sentiment analysis, and they are used for sentiment analysis in the English language without further tuning. The tokenization of Chinese reviews involves Chinese word segmentation.

Usually, if the semantic orientation value of a review is less than 0, the review is labeled as negative, otherwise, the review is labeled as positive.

3.4 Ensemble Combination

After obtaining the set of semantic orientation values $F_{SO}=\{f^k_{SO}(rev^k) \mid 0 \le k \le p\}$ by using the semantic oriented approach, where p is the number of English translations for each Chinese review, we exploit the following ensemble methods for deriving a new semantic orientation value $f_{SO}^{Ensemble}(rev^0)$:

1) Average

It is the most intuitive combination method and the new value is the average of the values in F_{SO}:

$$f_{SO}^{Ensemble}(rev^0) = \frac{\sum_{k=0}^{p} f_{SO}^{k}(rev^k)}{p+1}$$

Note that after the new value of a review is obtained, the polarity tag of the review is assigned in the same way as described in Section 3.3.

2) Weighted Average

This combination method improves the average combination method by associating each individual value with a weight, indicating the relative confidence in the value.

$$f_{SO}^{Ensemble}(rev^0) = \sum_{k=0}^{p} \lambda_k f_{SO}^{k}(rev^k)$$

where $\lambda_k \in [0, 1]$ is the weight associated with $f^k_{SO}(rev^k)$. The weights can be set in the following two ways:

Weighting Scheme1: The weight of $f^k_{SO}(rev^k)$ is set to the accuracy of the individual analysis in the *k*th language.

Weighting Scheme2: The weight of $f^k_{SO}(rev^k)$ is set to be the maximal correlation coefficient between the analysis results in the *k*th language and the analysis results in any other language. The underlying idea is that if the analysis results in one language are highly consistent with the analysis results in another language, the results are deemed to be more reliable. Given two lists of semantic values for all reviews, we use the Pearson's correlation coefficient to measure the correlation between them. The weight associated with function $f^k_{SO}(rev^k)$ is then defined as the maximal Pearson's correlation coefficient between the reviews' values in the *k*th language and the reviews' values in any other language.

3) Max

The new value is the maximum value in F_{SO}:

$$f_{SO}^{Ensemble}(rev^0) = \max\left\{f_{SO}^k(rev^k) \mid 0 \le k \le p\right\}$$

4) Min

The new value is the minimum value in F_{SO}:

$$f_{SO}^{Ensemble}(rev^0) = \min\left\{f_{SO}^k(rev^k) \mid 0 \le k \le p\right\}$$

5) Average Max&Min

The new value is the average of the maximum value and the minimum value in F_{SO}:

$$f_{SO}^{Ensemble}(rev^0) = \frac{\max\left\{f_{SO}^k(rev^k) \mid 0 \le k \le p\right\} + \min\left\{f_{SO}^k(rev^k) \mid 0 \le k \le p\right\}}{2}$$

6) Majority Voting

This combination method relies on the final polarity tags, instead of the semantic orientation values. A review can obtain p+1 polarity tags based on the individual analysis results in the p+1 languages. The polarity tag receiving more votes is chosen as the final polarity tag of the review.

4 Empirical Evaluation

4.1 Dataset and Evaluation Metrics

In order to assess the performance of the proposed approach, we collected 1000 product reviews from a popular Chinese IT product web site-IT168[10]. The reviews were posted by users and they focused on such products as mp3 players, mobile phones, digital camera and laptop computers. Users usually selected for each review an icon indicating "postive" or "negative". The reviews were first categorized into positive and negative classes according to the associated icon. The polarity labels for the reviews were then checked by subjects. Finally, the dataset contained 886 product reviews with accurate polarity labels. All the 886 reviews were used as test set.

We used the standard precision, recall and F-measure to measure the performance of positive and negative class, respectively, and used the MacroF measure and accuracy metric to measure the overall performance of the system. The metrics are defined the same as in general text categorization.

4.2 Individual Analysis Results

In this section, we investigate the following individual sentiment analysis results in each specified language:

CN: This method uses only Chinese lexicons to analyze Chinese reviews;

GoogleEN: This method uses only English lexicons to analyze English reviews translated by *GoogleTrans*;

YahooEN: This method uses only English lexicons to analyze English reviews translated by *YahooTrans*;

DictEN: This method uses only English lexicons to analyze English reviews translated by *DictTrans*;

In addition to the above methods for using English resources, the lexicon-based method investigated in Mihalcea et al. (2007) can also use English resources by directly projecting English lexicons into Chinese lexicons. We use a large English-to-Chinese dictionary - LDC_EC_DIC2.0[11] with 110834 entries for projecting English lexicons into Chinese lexicons via one-to-one translation. Based on the generated Chinese lexicons, two other individual methods are investigated in the experiments:

CN2: This method uses only the generated Chinese Resources to analyze Chinese reviews.

CN3: This method combines the original Chinese lexicons and the generated Chinese lexicons and uses the extended lexicons to analyze Chinese reviews.

Table 1 provides the performance values of all the above individual methods. Seen from the table, the performances of **GoogleEN** and **YahooEN** are much better than the baseline **CN** method, and even the **DictEN** performs as well as **CN**. The results demonstrate that the use of English resources for sentiment analysis of translated English reviews is an effective way for Chinese sentiment analysis. We can also see that the English sentiment analysis performance relies positively on the translation performance, and **GoogleEN** performs the best while **DictEN** performs the worst, which is consistent with the fact the *GoogleTrans* is deemed the best of the three machine translation systems, while *DictTrans* is the weakest one.

Furthermore, the **CN** method outperforms the **CN2** and **CN3** methods, and the **CN2** method performs the worst, which shows that the generated Chinese lexicons do not give any contributions to the performance of Chinese sentiment analysis. We explain the results by the fact that the term-based one-to-one translation is inaccurate and the generated Chinese lexicons are not reliable. Overall, the

[10] http://www.it168.com

[11] http://projects.ldc.upenn.edu/Chinese/LDC_ch.htm

approach through cross-lingual lexicon translation does not work well for Chinese sentiment analysis in our experiments.

4.3 Ensemble Results

In this section, we first use the simple average ensemble method to combine different individual analysis results. Table 2 provides the performance values of the average ensemble results based on different individual methods.

Seen from Tables 1 and 2, almost all of the average ensembles outperforms the baseline **CN** method and the corresponding individual methods, which shows that each individual methods have their own evidences for sentiment analysis, and thus fusing the evidences together can improve performance. For the methods of **CN+GoogleEN, CN+YahooEN** and **CN+DictEN**, we can see the ensemble performance is not positively relying on the translation performance: **CN+YahooEN** performs better than **CN+GoogleEN**, and even **CN+DictEN** performs as well as **CN+GoogleEN**. The results show that the individual methods in the ensembles can complement each other, and even the combination of two weak individual methods can achieve good performance. However, the **DictEN** method is not effective when the ensemble methods have already included **GoogleEN** and **YahooEN**. Overall, the performances of the ensemble methods rely on the performances of the most effective constituent individual methods: the methods including both **GoogleEN** and **YahooEN** perform much better than other methods, and **CN+GoogleEN+YahooEN** performs the best out of all the methods.

We further show the results of four typical average ensembles by varying the combination weights. The combination weights are respectively specified as λ·**CN**+(1-λ)·**GoogleEN**, λ·**CN**+(1-λ)·**YahooEN**, λ·**CN**+(1-λ)·**DictEN**, λ_1·**CN**+λ_2·**GoogleEN**+(1-λ_1-λ_2)·**YahooEN**. The results over the MacroF metric are shown in Figures 3 and 4 respectively. We can see from the figures that **GoogleEN** and **YahooEN** are dominant factors in the ensemble methods.

We then investigate to use other ensemble methods introduced in Section 3.4 to combine the **CN**, **GoogleEN** and **YahooEN** methods. Table 3 gives the comparison results. The methods of "**Weighted Average1**" and "**Weighted Average2**" are two weighted average ensembles using the two weighing schemes, respectively. We can see that all the ensemble methods outperform the constituent individual method, while the two weighted average ensembles perform the best. The results further demonstrate the good effectiveness of the ensemble combination of individual analysis results for Chinese sentiment analysis.

Individual Method	Positive			Negative			Total	
	Precision	Recall	F-measure	Precision	Recall	F-measure	MacroF	Accuracy
CN	0.681	0.929	0.786	0.882	0.549	0.677	0.732	0.743
CN2	0.615	0.772	0.684	0.678	0.499	0.575	0.630	0.638
CN3	0.702	0.836	0.763	0.788	0.632	0.702	0.732	0.736
GoogleEN	0.764	0.914	0.832	0.888	0.708	0.787	**0.810**	**0.813**
YahooEN	0.763	0.871	0.814	0.844	0.720	0.777	0.795	0.797
DictEN	0.738	0.761	0.749	0.743	0.720	0.731	0.740	0.740

Table 1. Individual analysis results

Average Ensemble	Positive			Negative			Total	
	Precision	Recall	F-measure	Precision	Recall	F-measure	MacroF	Accuracy
GoogleEN+YahooEN	0.820	0.900	0.858	0.885	0.795	0.838	0.848	0.848
GoogleEN+YahooEN +DictEN	0.841	0.845	0.843	0.838	0.834	0.836	0.840	0.840
CN+GoogleEN	0.754	0.949	0.840	0.928	0.678	0.784	0.812	0.816
CN+YahooEN	0.784	0.925	0.848	0.904	0.736	0.811	0.830	0.832
CN+DictEN	0.790	0.867	0.827	0.847	0.761	0.801	0.814	0.815
CN+GoogleEN +YahooEN	0.813	0.927	0.866	0.911	0.779	0.840	**0.853**	**0.854**
CN+GoogleEN+ YahooEN+DictEN	0.831	0.891	0.860	0.878	0.811	0.843	0.852	0.852

Table 2. Average combination results

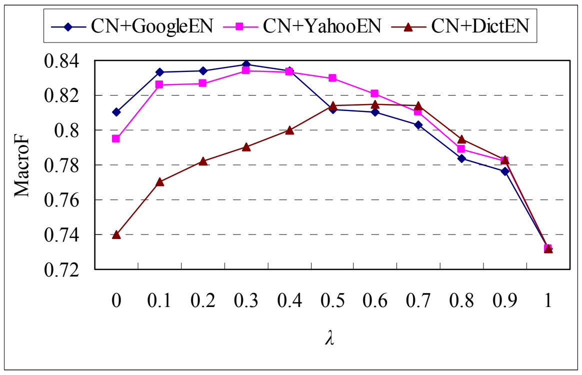


Figure 3. Ensemble performance vs. weight λ for λ˙**CN+**(1-λ)˙**GoogleEN/YahooEN/DictEN**

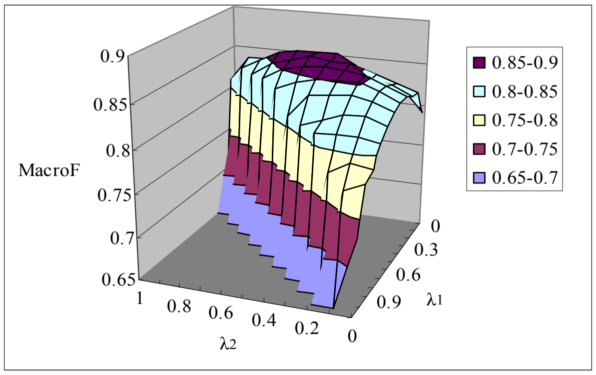


Figure 4. Ensemble performance vs. weights λ_1 and λ_2 for λ_1˙**CN+**λ_2˙**GoogleEN+**(1-λ_1-λ_2) ˙**YahooEN**

Ensemble Method	Positive			Negative			Total	
	Precision	Recall	F-measure	Precision	Recall	F-measure	MacroF	Accuracy
Average	0.813	0.927	0.866	0.911	0.779	0.840	0.853	0.854
Weighted Average1	0.825	0.922	0.871	0.908	0.798	0.849	**0.860**	**0.861**
Weighted Average2	0.822	0.922	0.869	0.908	0.793	0.847	0.858	0.859
Max	0.765	0.940	0.844	0.919	0.701	0.795	0.820	0.823
Min	0.901	0.787	0.840	0.805	0.910	0.854	0.847	0.848
Average Max&Min	0.793	0.936	0.859	0.918	0.747	0.824	0.841	0.843
Majority Voting	0.765	0.940	0.844	0.919	0.701	0.795	0.820	0.823

Table 3. Ensemble results for **CN & GoogleEN & YahooEN**

5 Conclusion and Future Work

This paper proposes a novel approach to use English sentiment resources for Chinese sentiment analysis by employing machine translation and ensemble techniques. Chinese reviews are translated into English reviews and the analysis results of both Chinese reviews and English reviews are combined to improve the overall accuracy. Experimental results demonstrate the encouraging performance of the proposed approach.

In future work, more additional English resources will be used to further improve the results. We will also apply the idea to supervised Chinese sentiment analysis.

Acknowledgments

This work was supported by the National Science Foundation of China (No.60703064), the Research Fund for the Doctoral Program of Higher Education of China (No.20070001059) and the National High Technology Research and Development Program of China (No.2008AA01Z421). We also thank the anonymous reviewers for their useful comments.

References

A. Andreevskaia and S. Bergler. 2008. When specialists and generalists work together: overcoming domain dependence in sentiment tagging. In *Proceedings of ACL-08: HLT*.

J. Blitzer, M. Dredze and F. Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: domain adaptation for sentiment classification. In *Proceedings of ACL2007*.

Y. Choi, E. Breck, and C. Cardie. 2006. Joint extraction of entities and relations for opinion recognition. In *Proc. EMNLP*.

A. Devitt and K. Ahmad. 2007. Sentiment polarity identification in financial news: a cohesion-based approach. In *Proceedings of ACL2007*.

K. Eguchi and V. Lavrenko. 2006. Sentiment retrieval using generative models. In *Proceedings of EMNLP*.

V. Hatzivassiloglou and K. R. McKeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of EACL*.

V. Hatzivassiloglon and J. M. Wiebe. 2000. Effects of adjective orientation and gradability on sentence subjectivity. In *Proceedings of COLING*.

K. Hiroshi, N. Tetsuya and W. Hideo. 2004. Deeper sentiment analysis using machine translation technology. In *Proceedings of COLING*.

N. Kaji and M. Kitsuregawa. 2007. Building lexicon for sentiment analysis from massive collection of HTML documents. In *Proceedings of EMNLP-CONLL*.

A. Kennedy and D. Inkpen. 2006. Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence*, 22(2):110-125.

S.-M. Kim and E. Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of COLING*.

L.-W. Ku, Y.-T. Liang and H.-H. Chen. 2006. Opinion extraction, summarization and tracking in news and blog corpora. In *Proceedings of AAAI*.

J. Li and M. Sun. 2007. Experimental study on sentiment classification of Chinese review using machine learning techniques. In *Proceeding of IEEE-NLPKE2007*.

B. Liu, M. Hu and J. Cheng. 2005. Opinion observer: Analyzing and comparing opinions on the web. In *Proceedings of WWW*.

R. McDonald, K. Hannan, T. Neylon, M. Wells and J. Reynar. 2007. Structured models for fine-to-coarse sentiment analysis. In *Proceedings of ACL2007*.

R. Mihalcea, C. Banea and J. Wiebe. 2007. Learning multilingual subjective language via cross-lingual projections. In *Proceedings of ACL*.

T. Mullen and N. Collier. 2004. Sentiment analysis using support vector machines with diverse information sources. In *Proceedings of EMNLP*.

B. Pang, L. Lee and S. Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP*.

B. Pang and L. Lee. 2004. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of ACL*.

A. –M. Popescu and O. Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of EMNLP*.

J. Read. 2005. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *Proceedings of ACL*.

E. Riloff and J. Wiebe 2003. Learning extraction patterns for subjective expressions. In *Proceedings of EMNLP2003*.

P. J. Stone, D. C. Dunphy, M. S. Smith, D. M. Ogilvie and associates. 1966. The General Inquirer: a computer approach to content analysis. The MIT Press.

H. Takamura, T. Inui and M. Okumura. 2005. Extracting semantic orientation of words using spin model. In *Proceedings of ACL*.

I. Titov and R. McDonald. 2008. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of ACL-08: HLT*.

B. K. Y. Tsou, R. W. M. Yuen, O. Y. Kwong, T. B. Y. La and W. L. Wong. 2005. Polarity classification of celebrity coverage in the Chinese press. In *Proceedings of International Conference on Intelligence Analysis*.

P. Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In Proceedings of ACL.

S. Wang, Y. Wei, D. Li, W. Zhang and W. Li. 2007. A hybrid method of feature selection for Chinese text sentiment classification. In *Proceeding of IEEE-FSKD2007*.

T. Wilson, J. Wiebe and P. Hoffmann. 2005a. Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. In *Proceedings of HLT/EMNLP2005*, Vancouver, Canada.

T. Wilson, P. Hoffmann, S. Somasundaran, J. Kessler, J. Wiebe, Y. Choi, C. Cardie, E. Riloff, S. Patwardhan. 2005b. OpinionFinder: a system for subjectivity analysis. In *Proceedings of HLP/EMNLP on Interactive Demonstrations*.

J. Yao, G. Wu, J. Liu and Y. Zheng. 2006. Using bilingual lexicon to judge sentiment orientation of Chinese words. In *Proceedings of IEEE CIT2006*.

Q. Ye, W. Shi and Y. Li. 2006. Sentiment classification for movie reviews in Chinese by improved semantic oriented approach. In *Proceedings of 39th Hawaii International Conference on System Sciences*.

H. Yu and V. Hatzivassiloglou. 2003. Towards answering opinion questions: separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of EMNLP2003*.

A Tale of Two Parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search

Yue Zhang and **Stephen Clark**
Oxford University Computing Laboratory
Wolfson Building, Parks Road
Oxford OX1 3QD, UK
{yue.zhang,stephen.clark}@comlab.ox.ac.uk

Abstract

Graph-based and transition-based approaches to dependency parsing adopt very different views of the problem, each view having its own strengths and limitations. We study both approaches under the framework of beam-search. By developing a graph-based and a transition-based dependency parser, we show that a beam-search decoder is a competitive choice for both methods. More importantly, we propose a beam-search-based parser that combines both graph-based and transition-based parsing into a single system for training and decoding, showing that it outperforms both the pure graph-based and the pure transition-based parsers. Testing on the English and Chinese Penn Treebank data, the combined system gave state-of-the-art accuracies of 92.1% and 86.2%, respectively.

1 Introduction

Graph-based (McDonald et al., 2005; McDonald and Pereira, 2006; Carreras et al., 2006) and transition-based (Yamada and Matsumoto, 2003; Nivre et al., 2006) parsing algorithms offer two different approaches to data-driven dependency parsing. Given an input sentence, a graph-based algorithm finds the highest scoring parse tree from all possible outputs, scoring each complete tree, while a transition-based algorithm builds a parse by a sequence of actions, scoring each action individually.

The terms "graph-based" and "transition-based" were used by McDonald and Nivre (2007) to describe the difference between MSTParser (McDonald and Pereira, 2006), which is a graph-based parser with an exhaustive search decoder, and MaltParser (Nivre et al., 2006), which is a transition-based parser with a greedy search decoder. In this paper, we do not differentiate graph-based and transition-based parsers by their search algorithms: a graph-based parser can use an approximate decoder while a transition-based parser is not necessarily deterministic. To make the concepts clear, we classify the two types of parser by the following two criteria:

1. whether or not the outputs are built by explicit transition-actions, such as "Shift" and "Reduce";
2. whether it is dependency graphs or transition-actions that the parsing model assigns scores to.

By this classification, beam-search can be applied to both graph-based and transition-based parsers.

Representative of each method, MSTParser and MaltParser gave comparable accuracies in the CoNLL-X shared task (Buchholz and Marsi, 2006). However, they make different types of errors, which can be seen as a reflection of their theoretical differences (McDonald and Nivre, 2007). MSTParser has the strength of exact inference, but its choice of features is constrained by the requirement of efficient dynamic programming. MaltParser is deterministic, yet its comparatively larger feature range is an advantage. By comparing the two, three interesting research questions arise: (1) how to increase the flexibility in defining features for graph-based parsing; (2) how to add search to transition-based parsing; and (3) how to combine the two parsing approaches so that the strengths of each are utilized.

In this paper, we study these questions under one framework: beam-search. Beam-search has been successful in many NLP tasks (Koehn et al., 2003;

```
Inputs: training examples $(x_i, y_i)$
Initialization: set $\vec{w} = 0$
Algorithm:
 // R training iterations; N examples
 for $t = 1..R$, $i = 1..N$:
     $z_i = \arg\max_{y \in \text{GEN}(x_i)} \Phi(y) \cdot \vec{w}$
     if $z_i \neq y_i$:
         $\vec{w} = \vec{w} + \Phi(y_i) - \Phi(z_i)$
Outputs: $\vec{w}$
```

Figure 1: The perceptron learning algorithm

Collins and Roark, 2004), and can achieve accuracy that is close to exact inference. Moreover, a beam-search decoder does not impose restrictions on the search problem in the way that an exact inference decoder typically does, such as requiring the "optimal subproblem" property for dynamic programming, and therefore enables a comparatively wider range of features for a statistical system.

We develop three parsers. Firstly, using the same features as MSTParser, we develop a graph-based parser to examine the accuracy loss from beam-search compared to exact-search, and the accuracy gain from extra features that are hard to encode for exact inference. Our conclusion is that beam-search is a competitive choice for graph-based parsing. Secondly, using the transition actions from MaltParser, we build a transition-based parser and show that search has a positive effect on its accuracy compared to deterministic parsing. Finally, we show that by using a beam-search decoder, we are able to combine graph-based and transition-based parsing into a single system, with the combined system significantly outperforming each individual system. In experiments with the English and Chinese Penn Treebank data, the combined parser gave 92.1% and 86.2% accuracy, respectively, which are comparable to the best parsing results for these data sets, while the Chinese accuracy outperforms the previous best reported by 1.8%. In line with previous work on dependency parsing using the Penn Treebank, we focus on projective dependency parsing.

2 The graph-based parser

Following MSTParser (McDonald et al., 2005; McDonald and Pereira, 2006), we define the graph-

```
Variables: $agenda$ – the beam for state items
           $item$ – partial parse tree
           $output$ – a set of output items
           $index, prev$ – word indexes
Input: $x$ – POS-tagged input sentence.
Initialization: $agenda$ = [""]
Algorithm:
for $index$ in 1..$x$.length():
    clear $output$
    for $item$ in $agenda$:
        // for all $prev$ words that can be linked with
        // the current word at $index$
        $prev = index - 1$
        while $prev \neq 0$: // while $prev$ is valid
            // add link making $prev$ parent of $index$
            $newitem = item$ // duplicate $item$
            $newitem$.link($prev$, $index$)  // modify
            $output$.append($newitem$)  // record
            // if $prev$ does not have a parent word,
            // add link making $index$ parent of $prev$
            if $item$.parent($prev$) == 0:
                $item$.link($index$, $prev$)  // modify
                $output$.append($item$)  // record
            $prev$ = the index of the first word before
                     $prev$ whose parent does not exist
                     or is on its left; 0 if no match
    clear $agenda$
    put the best items from $output$ to $agenda$
Output: the best item in $agenda$
```

Figure 2: A beam-search decoder for graph-based parsing, developed from the deterministic Covington algorithm for projective parsing (Covington, 2001).

based parsing problem as finding the highest scoring tree y from all possible outputs given an input x:

$$F(x) = \underset{y \in \text{GEN}(x)}{\arg\max}\, \text{Score}(y)$$

where GEN(x) denotes the set of possible parses for the input x. To repeat our earlier comments, in this paper we do not consider the method of finding the $\arg\max$ to be part of the definition of graph-based parsing, only the fact that the dependency graph itself is being scored, and factored into scores attached to the dependency links.

The score of an output parse y is given by a linear model:

$$\text{Score}(y) = \Phi(y) \cdot \vec{w}$$

where $\Phi(y)$ is the global feature vector from y and $\vec{w}$ is the weight vector of the model.

We use the discriminative perceptron learning algorithm (Collins, 2002; McDonald et al., 2005) to train the values of $\vec{w}$. The algorithm is shown in Figure 1. Averaging parameters is a way to reduce overfitting for perceptron training (Collins, 2002), and is applied to all our experiments.

While the MSTParser uses exact-inference (Eisner, 1996), we apply beam-search to decoding. This is done by extending the deterministic Covington algorithm for projective dependency parsing (Covington, 2001). As shown in Figure 2, the decoder works incrementally, building a state item (i.e. partial parse tree) word by word. When each word is processed, links are added between the current word and its predecessors. Beam-search is applied by keeping the B best items in the agenda at each processing stage, while partial candidates are compared by scores from the graph-based model, according to partial graph up to the current word.

Before decoding starts, the agenda contains an empty sentence. At each processing stage, existing partial candidates from the agenda are extended in all possible ways according to the Covington algorithm. The top B newly generated candidates are then put to the agenda. After all input words are processed, the best candidate output from the agenda is taken as the final output.

The projectivity of the output dependency trees is guaranteed by the incremental Covington process. The time complexity of this algorithm is $O(n^2)$, where n is the length of the input sentence.

During training, the "early update" strategy of Collins and Roark (2004) is used: when the correct state item falls out of the beam at any stage, parsing is stopped immediately, and the model is updated using the current best partial item. The intuition is to improve learning by avoiding irrelevant information: when all the items in the current agenda are incorrect, further parsing steps will be irrelevant because the correct partial output no longer exists in the candidate ranking.

Table 1 shows the feature templates from the MSTParser (McDonald and Pereira, 2006), which are defined in terms of the context of a word, its parent and its sibling. To give more templates, features from templates 1 – 5 are also conjoined with the link direction and distance, while features from template 6 are also conjoined with the direction and distance between the child and its sibling. Here "distance" refers to the difference between word indexes. We apply all these feature templates to the graph-based parser. In addition, we define two extra feature templates (Table 2) that capture information about grandchildren and arity (i.e. the number of children to the left or right). These features are not conjoined with information about direction and distance. They are difficult to include in an efficient dynamic programming decoder, but easy to include in a beam-search decoder.

1	Parent word (P)	Pw; Pt; Pwt
2	Child word (C)	Cw; Ct; Cwt
3	P and C	PwtCwt; PwtCw; PwCwt; PwtCt; PtCwt; PwCw; PtCt
4	A tag Bt between P, C	PtBtCt
5	Neighbour words of P, C, left (PL/CL) and right (PR/CR)	PtPLtCtCLt; PtPLtCtCRt; PtPRtCtCLt; PtPRtCtCRt; PtPLtCLt; PtPLtCRt; PtPRtCLt; PtPRtCRt; PLtCtCLt; PLtCtCRt; PRtCtCLt; PRtCtCRt; PtCtCLt; PtCtCRt; PtPLtCt; PtPRtCt
6	sibling (S) of C	CwSw; CtSt; CwSt; CtSw; PtCtSt;

Table 1: Feature templates from MSTParser
w – word; t – POS-tag.

1	leftmost (CLC) and rightmost (CRC) children of C	PtCtCLCt; PtCtCRCt
2	left (la) and right (ra) arity of P	Ptla; Ptra; Pwtla; Pwtra

Table 2: Additional feature templates for the graph-based parser

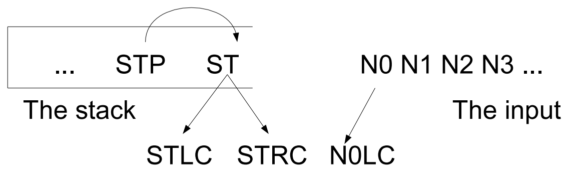


Figure 3: Feature context for the transition-based algorithm

3 The transition-based parser

We develop our transition-based parser using the transition model of the MaltParser (Nivre et al., 2006), which is characterized by the use of a stack and four transition actions: Shift, ArcRight, ArcLeft and Reduce. An input sentence is processed from left to right, with an index maintained for the current word. Initially empty, the stack is used throughout the parsing process to store unfinished words, which are the words before the current word that may still be linked with the current or a future word.

The Shift action pushes the current word to the stack and moves the current index to the next word. The ArcRight action adds a dependency link from the stack top to the current word (i.e. the stack top becomes the parent of the current word), pushes the current word on to the stack, and moves the current index to the next word. The ArcLeft action adds a dependency link from the current word to the stack top, and pops the stack. The Reduce action pops the stack. Among the four transition actions, Shift and ArcRight push a word on to the stack while ArcLeft and Reduce pop the stack; Shift and ArcRight read the next input word while ArcLeft and ArcRight add a link to the output. By repeated application of these actions, the parser reads through the input and builds a parse tree.

The MaltParser works deterministically. At each step, it makes a single decision and chooses one of the four transition actions according to the current context, including the next input words, the stack and the existing links. As illustrated in Figure 3, the contextual information consists of the top of stack (ST), the parent (STP) of ST, the leftmost (STLC) and rightmost child (STRC) of ST, the current word (N0), the next three words from the input (N1, N2, N3) and the leftmost child of N0 (N0LC). Given the context s, the next action T is decided as follows:

$$T(s) = \operatorname*{arg\,max}_{T \in \text{ACTION}} \text{Score}(T, s)$$

where ACTION $=$ $\{$Shift, ArcRight, ArcLeft, Reduce$\}$.

One drawback of deterministic parsing is error propagation, since once an incorrect action is made, the output parse will be incorrect regardless of the subsequent actions. To reduce such error propagation, a parser can keep track of multiple candidate outputs and avoid making decisions too early. Suppose that the parser builds a set of candidates GEN(x) for the input x, the best output $F(x)$ can be decided by considering all actions:

$$F(x) = \operatorname*{arg\,max}_{y \in \text{GEN}(x)} \textstyle\sum_{T' \in \text{act}(y)} \text{Score}(T', s_{T'})$$

Here T' represents one action in the sequence (act(y)) by which y is built, and $s_{T'}$ represents the corresponding context when T' is taken.

Our transition-based algorithm keeps B different sequences of actions in the agenda, and chooses the one having the overall best score as the final parse. Pseudo code for the decoding algorithm is shown in Figure 4. Here each state item contains a partial parse tree as well as a stack configuration, and state items are built incrementally by transition actions. Initially the stack is empty, and the agenda contains an empty sentence. At each processing stage, one transition action is applied to existing state items as a step to build the final parse. Unlike the MaltParser, which makes a decision at each stage, our transition-based parser applies all possible actions to each existing state item in the agenda to generate new items; then from all the newly generated items, it takes the B with the highest overall score and puts them onto the agenda. In this way, some ambiguity is retained for future resolution.

Note that the number of transition actions needed to build different parse trees can vary. For example, the three-word sentence “A B C” can be parsed by the sequence of three actions “Shift ArcRight ArcRight” (B modifies A; C modifies B) or the sequence of four actions “Shift ArcLeft Shift ArcRight” (both A and C modifies B). To ensure that all final state items are built by the same number of transition actions, we require that the final state

```
Variables: $agenda$ – the beam for state items
           $item$ – (partial tree, stack config)
           $output$ – a set of output items
           $index$ – iteration index
Input: $x$ – POS-tagged input sentence.
Initialization: $agenda$ = [(“”, [])]
Algorithm:
for $index$ in 1 .. $2 \times x$.length() $-1$:
    clear $output$
    for $item$ in $agenda$:
        // when all input words have been read, the
        // parse tree has been built; only pop.
        if $item$.length() == $x$.length():
            if $item$.stacksize() $> 1$:
                $item$.Reduce()
                $output$.append($item$)
        // when some input words have not been read
        else:
            if $item$.lastaction() $\neq$ Reduce:
                $newitem = item$
                $newitem$.Shift()
                $output$.append($newitem$)
            if $item$.stacksize() $> 0$:
                $newitem = item$
                $newitem$.ArcRight()
                $output$.append($newitem$)
                if ($item$.parent($item$.stacktop())==0):
                    $newitem = item$
                    $newitem$.ArcLeft()
                    $output$.append($newitem$)
                else:
                    $newitem = item$
                    $newitem$.Reduce()
                    $output$.append($newitem$)
    clear $agenda$
    transfer the best items from $output$ to $agenda$
Output: the best item in $agenda$
```

Figure 4: A beam-search decoding algorithm for transition-based parsing

items must 1) have fully-built parse trees; and 2) have only one root word left on the stack. In this way, popping actions should be made even after a complete parse tree is built, if the stack still contains more than one word.

Now because each word excluding the root must be pushed to the stack once and popped off once during the parsing process, the number of actions needed to parse a sentence is always $2n-1$, where n is the length of the sentence. Therefore, the decoder has linear time complexity, given a fixed beam size. Because the same transition actions as the MaltParser are used to build each item, the projectivity of the output dependency tree is ensured.

We use a linear model to score each transition action, given a context:

$$Score(T, s) = \Phi(T, s) \cdot \vec{w}$$

$\Phi(T, s)$ is the feature vector extracted from the action T and the context s, and $\vec{w}$ is the weight vector. Features are extracted according to the templates shown in Table 3, which are based on the context in Figure 3. Note that our feature definitions are similar to those used by MaltParser, but rather than using a kernel function with simple features (e.g. STw,

```
Inputs: training examples $(x_i, y_i)$
Initialization: set $\vec{w} = 0$
Algorithm:
// R training iterations; N examples
for $t = 1..R$, $i = 1..N$:
    $z_i = \arg\max_{y \in \text{GEN}(x_i)} \sum_{T' \in \text{act}(y_i)} \Phi(T', c') \cdot \vec{w}$
    if $z_i \neq y_i$:
        $\vec{w} = \vec{w} + \sum_{T' \in \text{act}(y_i)} \Phi(T', c_{T'})$
                $- \sum_{T' \in \text{act}(z_i)} \Phi(T', c_{T'})$
Outputs: $\vec{w}$
```

Figure 5: the perceptron learning algorithm for the transition-based parser

1	stack top	STwt; STw; STt
2	current word	N0wt; N0w; N0t
3	next word	N1wt; N1w; N1t
4	ST and N0	STwtN0wt; STwtN0w; STwN0wt; STwtN0t; STtN0wt; STwN0w; STtN0t
5	POS bigram	N0tN1t
6	POS trigrams	N0tN1tN2t; STtN0tN1t; STPtSTtN0t; STtSTLCtN0t; STtSTRCtN0t; STtN0tN0LCt
7	N0 word	N0wN1tN2t; STtN0wN1t; STPtSTtN0w; STtSTLCtN0w; STtSTRCtN0w; STtN0wN0LCt

Table 3: Feature templates for the transition-based parser
w – word; t – POS-tag.

N0t, but not STwt or STwN0w), we combine features manually.

As with the graph-based parser, we use the discriminative perceptron (Collins, 2002) to train the transition-based model (see Figure 5). It is worth noticing that, in contrast to MaltParser, which trains each action decision individually, our training algorithm globally optimizes all action decisions for a parse. Again, "early update" and averaging parameters are applied to the training process.

4 The combined parser

The graph-based and transition-based approaches adopt very different views of dependency parsing. McDonald and Nivre (2007) showed that the MSTParser and MaltParser produce different errors. This observation suggests a combined approach: by using both graph-based information and transition-based information, parsing accuracy can be improved.

The beam-search framework we have developed facilitates such a combination. Our graph-based and transition-based parsers share many similarities. Both build a parse tree incrementally, keeping an agenda of comparable state items. Both rank state items by their current scores, and use the averaged perceptron with early update for training. The key differences are the scoring models and incremental parsing processes they use, which must be addressed when combining the parsers.

Firstly, we combine the graph-based and the transition-based score models simply by summation. This is possible because both models are global and linear. In particular, the transition-based model can be written as:

$$\begin{aligned}\text{Score}_{\text{T}}(y) &= \textstyle\sum_{T' \in \text{act}(y)} \text{Score}(T', s_{T'}) \\ &= \textstyle\sum_{T' \in \text{act}(y)} \Phi(T', s_{T'}) \cdot \vec{w_{\text{T}}} \\ &= \vec{w_{\text{T}}} \cdot \textstyle\sum_{T' \in \text{act}(y)} \Phi(T', s_{T'})\end{aligned}$$

If we take $\sum_{T' \in \text{act}(y)} \Phi(T', s_{T'})$ as the global feature vector $\Phi_{\text{T}}(y)$, we have:

$$Score_{\text{T}}(y) = \Phi_{\text{T}}(y) \cdot \vec{w_{\text{T}}}$$

which has the same form as the graph-based model:

$$Score_{\text{G}}(y) = \Phi_{\text{G}}(y) \cdot \vec{w_{\text{G}}}$$

	Sections	Sentences	Words
Training	2–21	39,832	950,028
Dev	22	1,700	40,117
Test	23	2,416	56,684

Table 4: The training, development and test data from PTB

We therefore combine the two models to give:

$$\begin{aligned}Score_{\text{C}}(y) &= Score_{\text{G}}(y) + Score_{\text{T}}(y) \\ &= \Phi_{\text{G}}(y) \cdot \vec{w_{\text{G}}} + \Phi_{\text{T}}(y) \cdot \vec{w_{\text{T}}}\end{aligned}$$

Concatenating the feature vectors $\Phi_{\text{G}}(y)$ and $\Phi_{\text{T}}(y)$ to give a global feature vector $\Phi_{\text{C}}(y)$, and the weight vectors $\vec{w_{\text{G}}}$ and $\vec{w_{\text{T}}}$ to give a weight vector $\vec{w_{\text{C}}}$, the combined model can be written as:

$$Score_{\text{C}}(y) = \Phi_{\text{C}}(y) \cdot \vec{w_{\text{C}}}$$

which is a linear model with exactly the same form as both sub-models, and can be trained with the perceptron algorithm in Figure 1. Because the global feature vectors from the sub models are concatenated, the feature set for the combined model is the union of the sub model feature sets.

Second, the transition-based decoder can be used for the combined system. Both the graph-based decoder in Figure 2 and the transition-based decoder in Figure 4 construct a parse tree incrementally. However, the graph-based decoder works on a per-word basis, adding links without using transition actions, and so is not appropriate for the combined model. The transition-based algorithm, on the other hand, uses state items which contain partial parse trees, and so provides all the information needed by the graph-based parser (i.e. dependency graphs), and hence the combined system.

In summary, we build the combined parser by using a global linear model, the union of feature templates and the decoder from the transition-based parser.

5 Experiments

We evaluate the parsers using the English and Chinese Penn Treebank corpora. The English data is prepared by following McDonald et al. (2005). Bracketed sentences from the Penn Treebank (PTB) 3 are split into training, development and test sets

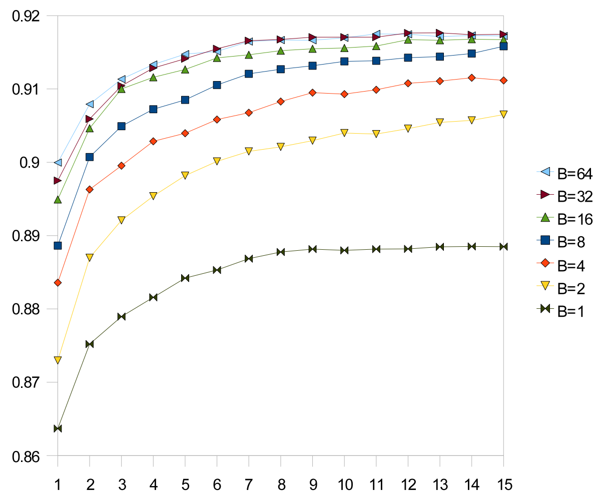


Figure 6: The influence of beam size on the transition-based parser, using the development data
X-axis: number of training iterations
Y-axis: word precision

as shown in Table 4, and then translated into dependency structures using the head-finding rules from Yamada and Matsumoto (2003).

Before parsing, POS tags are assigned to the input sentence using our reimplementation of the POS-tagger from Collins (2002). Like McDonald et al. (2005), we evaluate the parsing accuracy by the precision of lexical heads (the percentage of input words, excluding punctuation, that have been assigned the correct parent) and by the percentage of complete matches, in which all words excluding punctuation have been assigned the correct parent.

5.1 Development experiments

Since the beam size affects all three parsers, we study its influence first; here we show the effect on the transition-based parser. Figure 6 shows different accuracy curves using the development data, each with a different beam size B. The X-axis represents the number of training iterations, and the Y-axis the precision of lexical heads.

The parsing accuracy generally increases as the beam size increases, while the quantity of increase becomes very small when B becomes large enough. The decoding times after the first training iteration are 10.2s, 27.3s, 45.5s, 79.0s, 145.4s, 261.3s and 469.5s, respectively, when $B = 1, 2, 4, 8, 16, 32, 64$.

	Word	Complete
MSTParser 1	90.7	36.7
Graph [M]	91.2	40.8
Transition	91.4	41.8
Graph [MA]	91.4	42.5
MSTParser 2	91.5	42.1
Combined [TM]	92.0	45.0
Combined [TMA]	92.1	45.4

Table 5: Accuracy comparisons using PTB 3

In the rest of the experiments, we set $B = 64$ in order to obtain the highest possible accuracy.

When $B = 1$, the transition-based parser becomes a deterministic parser. By comparing the curves when $B = 1$ and $B = 2$, we can see that, while the use of search reduces the parsing speed, it improves the quality of the output parses. Therefore, beam-search is a reasonable choice for transition-based parsing.

5.2 Accuracy comparisons

The test accuracies are shown in Table 5, where each row represents a parsing model. Rows "MSTParser 1/2" show the first-order (using feature templates 1 – 5 from Table 1) (McDonald et al., 2005) and second-order (using all feature templates from Table 1) (McDonald and Pereira, 2006) MSTParsers, as reported by the corresponding papers. Rows "Graph [M]" and "Graph [MA]" represent our graph-based parser using features from Table 1 and Table 1 + Table 2, respectively; row "Transition" represents our transition-based parser; and rows "Combined [TM]" and "Combined [TMA]" represent our combined parser using features from Table 3 + Table 1 and Table 3 + Table 1 + Table 2, respectively. Columns "Word" and "Complete" show the precision of lexical heads and complete matches, respectively.

As can be seen from the table, beam-search reduced the head word accuracy from $91.5\%/42.1\%$ ("MSTParser 2") to $91.2\%/40.8\%$ ("Graph [M]") with the same features as exact-inference. However, with only two extra feature templates from Table 2, which are not conjoined with direction or distance information, the accuracy is improved to $91.4\%/42.5\%$ ("Graph [MA]"). This improvement can be seen as a benefit of beam-search, which allows the definition of more global features.

	Sections	Sentences	Words
Training	001–815; 1001–1136	16,118	437,859
Dev	886–931; 1148–1151	804	20,453
Test	816–885; 1137–1147	1,915	50,319

Table 6: Training, development and test data from CTB

	Non-root	Root	Comp.
Graph [MA]	83.86	71.38	29.82
Duan 2007	84.36	73.70	32.70
Transition	84.69	76.73	32.79
Combined [TM]	86.13	77.04	35.25
Combined [TMA]	86.21	76.26	34.41

Table 7: Test accuracies with CTB 5 data

The combined parser is tested with various sets of features. Using only graph-based features in Table 1, it gave 88.6% accuracy, which is much lower than 91.2% from the graph-based parser using the same features ("Graph [M]"). This can be explained by the difference between the decoders. In particular, the graph-based model is unable to score the actions "Reduce" and "Shift", since they do not modify the parse tree. Nevertheless, the score serves as a reference for the effect of additional features in the combined parser.

Using both transition-based features and graph-based features from the MSTParser ("Combined [TM]"), the combined parser achieved 92.0% per-word accuracy, which is significantly higher than the pure graph-based and transition-based parsers. Additional graph-based features further improved the accuracy to $92.1\%/45.5\%$, which is the best among all the parsers compared.[1]

5.3 Parsing Chinese

We use the Penn Chinese Treebank (CTB) 5 for experimental data. Following Duan et al. (2007), we split the corpus into training, development and test data as shown in Table 6, and use the head-finding rules in Table 8 in the Appendix to turn the bracketed sentences into dependency structures. Most of the head-finding rules are from Sun and Jurafsky (2004), while we added rules to handle NN and FRAG, and a default rule to use the rightmost node as the head for the constituent that are not listed.

Like Duan et al. (2007), we use gold-standard POS-tags for the input. The parsing accuracy is evaluated by the percentage of non-root words that have been assigned the correct head, the percentage of correctly identified root words, and the percentage of complete matches, all excluding punctuation.

The accuracies are shown in Table 7. Rows "Graph [MA]", "Transition", "Combined [TM]" and "Combined [TMA]" show our models in the same way as for the English experiments from Section 5.2. Row "Duan 2007" represents the transition-based model from Duan et al. (2007), which applies beam-search to the deterministic model from Yamada and Matsumoto (2003), and achieved the previous best accuracy on the data.

Our observations on parsing Chinese are essentially the same as for English. Our combined parser outperforms both the pure graph-based and the pure transition-based parsers. It gave the best accuracy we are aware of for dependency parsing using CTB.

6 Related work

Our graph-based parser is derived from the work of McDonald and Pereira (2006). Instead of performing exact inference by dynamic programming, we incorporated the linear model and feature templates from McDonald and Pereira (2006) into our beam-search framework, while adding new global features. Nakagawa (2007) and Hall (2007) also showed the effectiveness of global features in improving the accuracy of graph-based parsing, using the approximate Gibbs sampling method and a reranking approach, respectively.

Our transition-based parser is derived from the deterministic parser of Nivre et al. (2006). We incorporated the transition process into our beam-search framework, in order to study the influence of search on this algorithm. Existing efforts to add search to deterministic parsing include Sagae

[1]A recent paper, Koo et al. (2008) reported parent-prediction accuracy of 92.0% using a graph-based parser with a different (larger) set of features (Carreras, 2007). By applying separate word cluster information, Koo et al. (2008) improved the accuracy to 93.2%, which is the best known accuracy on the PTB data. We excluded these from Table 5 because our work is not concerned with the use of such additional knowledge.

and Lavie (2006b), which applied best-first search to constituent parsing, and Johansson and Nugues (2006) and Duan et al. (2007), which applied beam-search to dependency parsing. All three methods estimate the probability of each transition action, and score a state item by the product of the probabilities of all its corresponding actions. But different from our transition-based parser, which trains all transitions for a parse globally, these models train the probability of each action separately. Based on the work of Johansson and Nugues (2006), Johansson and Nugues (2007) studied global training with an approximated large-margin algorithm. This model is the most similar to our transition-based model, while the differences include the choice of learning and decoding algorithms, the definition of feature templates and our application of the "early update" strategy.

Our combined parser makes the biggest contribution of this paper. In contrast to the models above, it includes both graph-based and transition-based components. An existing method to combine multiple parsing algorithms is the ensemble approach (Sagae and Lavie, 2006a), which was reported to be useful in improving dependency parsing (Hall et al., 2007). A more recent approach (Nivre and McDonald, 2008) combined MSTParser and MaltParser by using the output of one parser for features in the other. Both Hall et al. (2007) and Nivre and McDonald (2008) can be seen as methods to combine separately defined models. In contrast, our parser combines two components in a single model, in which all parameters are trained consistently.

7 Conclusion and future work

We developed a graph-based and a transition-based projective dependency parser using beam-search, demonstrating that beam-search is a competitive choice for both parsing approaches. We then combined the two parsers into a single system, using discriminative perceptron training and beam-search decoding. The appealing aspect of the combined parser is the incorporation of two largely different views of the parsing problem, thus increasing the information available to a single statistical parser, and thereby significantly increasing the accuracy. When tested using both English and Chinese dependency data, the combined parser was highly competitive compared to the best systems in the literature.

The idea of combining different approaches to the same problem using beam-search and a global model could be applied to other parsing tasks, such as constituent parsing, and possibly other NLP tasks.

Acknowledgements

This work is supported by the ORS and Clarendon Fund. We thank the anonymous reviewers for their detailed comments.

Appendix

Constituent	Rules
ADJP	r ADJP JJ AD; r
ADVP	r ADVP AD CS JJ NP PP P VA VV; r
CLP	r CLP M NN NP; r
CP	r CP IP VP; r
DNP	r DEG DNP DEC QP; r
DP	r M; l DP DT OD; l
DVP	r DEV AD VP; r
FRAG	r VV NR NN NT; r
IP	r VP IP NP; r
LCP	r LCP LC; r
LST	r CD NP QP; r
NP	r NP NN IP NR NT; r
NN	r NP NN IP NR NT; r
PP	l P PP; l
PRN	l PU; l
QP	r QP CLP CD; r
UCP	l IP NP VP; l
VCD	l VV VA VE; l
VP	l VE VC VV VNV VPT VRD VSB VCD VP; l
VPT	l VA VV; l
VRD	l VVI VA; l
VSB	r VV VE; r
default	r

Table 8: Head-finding rules to extract dependency data from CTB

References

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL*, pages 149–164, New York City, USA, June.

Xavier Carreras, Mihai Surdeanu, and Lluis Marquez. 2006. Projective dependency parsing with perceptron. In *Proceedings of CoNLL*, New York City, USA, June.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP/CoNLL*, pages 957–961, Prague, Czech Republic, June.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL*, pages 111–118, Barcelona, Spain, July.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*, pages 1–8, Philadelphia, USA, July.

Michael A. Covington. 2001. A fundamental algorithm for dependency parsing. In *Proceedings of the ACM Southeast Conference*, Athens, Georgia, March.

Xiangyu Duan, Jun Zhao, and Bo Xu. 2007. Probabilistic models for action-based chinese dependency parsing. In *Proceedings of ECML/ECPPKDD*, Warsaw, Poland, September.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING*, pages 340–345, Copenhagen, Denmark, August.

Johan Hall, Jens Nilsson, Joakim Nivre, Gülsen Eryigit, Beáta Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single malt or blended? a study in multilingual parser optimization. In *Proceedings of the CoNLL Shared Task Session of EMNLP/CoNLL*, pages 933–939, Prague, Czech Republic, June.

Keith Hall. 2007. K-best spanning tree parsing. In *Proceedings of ACL*, Prague, Czech Republic, June.

Richard Johansson and Pierre Nugues. 2006. Investigating multilingual dependency parsing. In *Proceedings of CoNLL*, pages 206–210, New York City, USA, June.

Richard Johansson and Pierre Nugues. 2007. Incremental dependency parsing using online learning. In *Proceedings of the CoNLL/EMNLP*, pages 1134–1138, Prague, Czech Republic.

Philip Koehn, Franz Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL/HLT*, Edmonton, Canada, May.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL/HLT*, pages 595–603, Columbus, Ohio, June.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of EMNLP/CoNLL*, pages 122–131, Prague, Czech Republic, June.

R McDonald and F Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *In Proc. of EACL*, pages 81–88, Trento, Italy, April.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*, pages 91–98, Ann Arbor, Michigan, June.

Tetsuji Nakagawa. 2007. Multilingual dependency parsing using global features. In *Proceedings of the CoNLL Shared Task Session of EMNLP/CoNLL*, pages 952–956, Prague, Czech Republic, June.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL/HLT*, pages 950–958, Columbus, Ohio, June.

Joakim Nivre, Johan Hall, Jens Nilsson, Gülşen Eryiğit, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of CoNLL*, pages 221–225, New York City, USA, June.

K Sagae and A Lavie. 2006a. Parser combination by reparsing. In *In Proc. HLT/NAACL*, pages 129–132, New York City, USA, June.

Kenji Sagae and Alon Lavie. 2006b. A best-first probabilistic shift-reduce parser. In *Proceedings of COLING/ACL (poster)*, pages 691–698, Sydney, Australia, July.

Honglin Sun and Daniel Jurafsky. 2004. Shallow semantic parsing of Chinese. In *Proceedings of NAACL/HLT*, Boston, USA, May.

H Yamada and Y Matsumoto. 2003. Statistical dependency analysis using support vector machines. In *Proceedings of IWPT*, Nancy, France, April.

Generalizing Local and Non-Local Word-Reordering Patterns for Syntax-Based Machine Translation

Bing Zhao
IBM T.J. Watson Research
Yorktown Heights, NY-10598
zhaob@us.ibm.com

Yaser Al-onaizan
IBM T.J. Watson Research
Yorktown Heights, NY-10598
onaizan@us.ibm.com

Abstract

Syntactic word reordering is essential for translations across different grammar structures between syntactically distant language-pairs. In this paper, we propose to embed local and non-local word reordering decisions in a synchronous context free grammar, and leverages the grammar in a chart-based decoder. Local word-reordering is effectively encoded in Hiero-like rules; whereas non-local word-reordering, which allows for long-range movements of syntactic chunks, is represented in tree-based reordering rules, which contain variables correspond to source-side syntactic constituents. We demonstrate how these rules are learned from parallel corpora. Our proposed shallow Tree-to-String rules show significant improvements in translation quality across different test sets.

1 Introduction

One of the main issues that a translator (human or machine) must address during the translation process is how to match the different word orders between the source language and the target language. Different language-pairs require different levels of word reordering. For example, when we translate between English and Spanish (or other Romance languages), most of the word reordering needed is local because of the shared syntactical features (e.g., Spanish noun modifier constructs are written in English as modifier noun). However, for syntactically distant language-pairs such as Chinese-English, long-range reordering is required where whole phrases are moved across the sentence.

The idea of "*syntactic cohesion*" (Fox, 2002) is characterized by its simplicity, which has attracted researchers for years. Previous works include several approaches of incorporating syntactic information to *preprocess* the source sentences to make them more like the target language in structure. Xia and McCord (2004) (Niessen and Ney, 2004; Collins et al., 2005) described approaches applied to language-pairs such as French-English and German-English. Later, Wang et al. (2007) presented specific rules to pre-order long-range movements of words, and improved the translations for Chinese-to-English. Overall, these works are similar, in that they design a few language-specific and linguistically motivated reordering rules, which are generally simple. The *eleven* rules described in Wang et al. (2007) are appealing, as they have rather simple structure, modeling only NP, VP and LCP via one-level sub-tree structure with two children, in the source parse-tree (a special case of ITG (Wu, 1997)). It effectively enhances the quality of the phrase-based translation of Chinese-to-English. One major weakness is that the reordering decisions were done in the preprocessing step, therefore rendering the decoding process unable to recover the reordering errors from the rules if incorrectly applied to. Also the reordering decisions are made without the benefits of additional models (e.g., the language models) that are typically used during decoding.

Another method to address the re-ordering problem in translation is the Hiero model proposed by Chiang (2005), in which a probabilistic synchronous context free grammar (PSCFG) was applied to guide the decoding. Hiero rules generalize phrase-pairs by introducing a single generic nonterminal (i.e., a variable) [X]. The combination of variables and lexicalized words in a Hiero rule nicely captures local word and phrase reordering (modeling an implicit reordering window of max-phrase length). These rules are then applied in a CYK-style decoder. In Hiero rules, any nested phrase-pair can be generalized as variables [X]. This usually leads to too many redundant translations, which worsens the *spurious ambiguities* (Chiang, 2005) problems for both decoding and optimization (i.e., parameter tuning). We found that*variables* (*nonterminal* [X]) in Hiero rules offer a generalization too coarse to improve the effectiveness of hierarchical models' performance.

We propose to enrich the variables in Hiero rules with additional source syntactic reordering information, in the form of shallow Tree-to-String syntactic structures. The syntactic information is represented by flat one-level sub-tree structures, with Hiero-like nonterminal variables at the leaf nodes. The syntactic rules, proposed in this paper, are composed of (possibly lexicalized) source treelets and target surface strings, with one or more variables that help capture local-reordering similar to the Hiero rules. Variables in a given rule are derived not only from the embedded aligned blocks (phrase-pairs), but also from the aligned source syntactic constituents. The aligned constituents, as in our empirical observations for Chinese-English, tend to move together in translations. The decoder is guided by these rules to reduce spurious derivations; the rules also constrain the exploration of the search space toward better translation quality and sometime improved speed by breaking long sentences into pieces. Overall, what we want is to enable the long-range reordering decisions to be local in a chart-based decoder.

To be more specific, we think the simple shallow syntactic structure is powerful enough for capturing the major structure-reordering patterns, such as NP, VP and LCP structures. We also use simple frequency-based feature functions, similar to the blocks used in phrase-based decoder, to further improve the rules' representation power. Overall, this enables us to avoid either a complex decoding process to generate the source parse tree, or difficult combinatorial optimizations for the feature functions associated with rules.

In Marton and Resnik (2008), hiero variables were disambiguated with additional binary feature functions, with their weights optimized in standard MER training. The combinatorial effects of the added feature functions can make the feature selection and optimization of the weights rather difficult. Since the grammar is essentially the same as the Hiero ones, a standard CYK decoder can be simply applied in their work. Word reordering can also be addressed via distortion models. Work in (Al-Onaizan and Kishore, 2006; Xiong et al., 2006; Zens et al., 2004; Kumar and Byrne, 2005; Tillmann and Zhang, 2005) modeled the limited information available at phrase-boundaries. Syntax-based approaches such as (Yamada and Knight, 2001; Graehl and Knight, 2004; Liu et al., 2006) heavily rely on the parse-tree to constrain the search space by assuming a strong mapping of structures across distant language-pairs. Their algorithms are also subject to parsers' performances to a larger extent, and have high complexity and less scalability in reality. In Liu et al. (2007), multi-level tree-structured rules were designed, which made the decoding process very complex, and auxiliary rules have to be designed and incorporated to shrink multiple source nonterminals into one target nonterminal. From our empirical observations, most of the time, however, the multi-level tree-structure is broken in the translation process, and POS tags are frequently distorted. Indeed, strictly following the source parse tree is usually not necessary, and maybe too expensive for the translation process.

The remainder of this paper is structured as follows: in section § 2, we define the notations in our synchronous context free grammar, in section § 3, the rule extractions are illustrated in details, in section § 4, the decoding process of applying these rules is described. Experiments in § 5 were carried out using GALE Dev07 datasets. Improved translation qualities were obtained by applying the proposed Tree-to-String rules. Conclusions and discussions are given in § 6.

2 Shallow Tree-to-String Rules

Our proposed rules are in the form of probabilistic synchronous context free grammar (PSCFG). We adopt the notations used in (Chiang, 2005). Let N be a set of nonterminals, a rule has the following form:

$$X \rightarrow < \ell; \gamma; \alpha; \sim; \bar{w} >, \qquad (1)$$

where X abstracts nonterminal symbols in N; $\gamma \in [N, V_S]^+$ is a sequence of one or more source [1] words (as in the vocabulary of V_S) and nonterminal symbols in N; $\alpha \in [N, V_T]^+$ is a sequence of one or more target words (in V_T) and nonterminals in N . $\sim$ is the one-to-one alignment of the nonterminals between γ and α; $\bar{w}$ contains non-negative weights associated with each rule; ℓ is a label-symbol specifying the root node of the source span covering γ. In our grammar, ℓ is one of the labels (e.g., NP) defined in the source treebank tagset (in our case UPenn Chinese tagset) indicating that the source span γ is rooted at ℓ. Additionally, a NULL tag Ø in ℓ denotes a flat structure of γ, in which no constituent structure was found to cover the span, and we need to back off to the normal Hiero-style rules. Our nonterminal symbols include the labels and the POS tags in the source parse trees.

In the following, we will illustrate the Tree-to-String rules we are proposing. At the same time, we will describe the extraction algorithm, with which we derive our rules from the word-aligned source-parsed parallel text. Our nonterminal set N is a reduced set of the treebank tagset (Xue et al., 2005). It consists of 17 unique labels.

The rules we extract belong to one of the following categories:

- γ contains only words, and ℓ is NULL; this corresponds to the general blocks used in phrase-based decoder (Och and Ney, 2004);
- γ contains words and variables of [X,0] and [X,1], and ℓ is NULL; this corresponds to the Hiero rules as in Chiang (2005);
- γ contains words and variables in the form of [X,TAG[2]], in which TAG is from the LDC tagset; this defines a well formed subtree, in which at least one child (constituent) is aligned to continuous target ngrams. If γ contains only variables from LDC tag set, this indicates all the constituents (children) in the subtree are aligned. This is a superset of rules generalizing those in Wang et al. (2007). If γ contains variables from POS tags, this essentially produces a superset of the monolingual side POS-based reordering rules explored in Tillmann (2008).

We focus on the third category — a syntactic label ℓ over the span of γ, indicating the covered source words consist of a linguistically well-defined phrase. ℓ together with γ define a tree-like structure: the root node is ℓ, and the aligned children are nonterminals in γ. The structure information is encoded in (ℓ, γ) pair-wise connections, and the variables keep the generalizations over atomic translation-pairs similar to Hiero models. When the rule is applied during decoding time, the labels, the tree-structure and the lexical items need to be all matched.

3 Learning and Applying Rules

A parser is assumed for the source language in the parallel data. In our case, a Chinese parser is applied for training and test data. A word alignment model is used to align the source words with the target words.

3.1 Extractions

Our rule extraction is a three-step process. First, traditional blocks (phrase-pairs) extraction is carried out. Secondly, Tree-to-String rules, are then extracted from the aligned blocks, of which the source side is covered by a complete subtree, with different permutations of the embedded aligned constituents, or partially lexicalized constituents. Otherwise, the Hiero-like rules will be extracted when there is no sub-tree structure identified, in our final step. Frequencies of extracted rules were counted to compute feature functions.

Figure 1-(a) shows that a subtree (with root at VP) is aligned to the English string. Considering the huge quantity of all the permutations of the aligned constituents under the tree, only part of the Tree-to-String rules extracted are shown in Figure 1-(c). The variables incorporate linguistic information in the assigned tag by the parser. When there is no aligned constituent for further generalization, the variables, defined in our grammar, back off to the Hiero-like ones without any label-identity information. One such example is in the rule "在 [X,0] 前 [X,VP] $\rightarrow$ [X,VP] before the [X,0]", in which the Hiero-style

[1]we use end-user terminologies for *source* and *target*.

[2]we index the tags for multiple occurrences in one rule

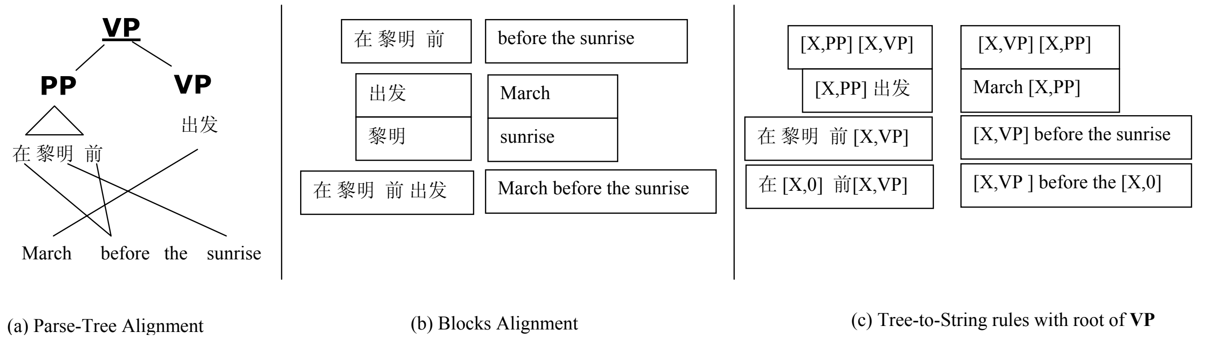


Figure 1: Example rules extracted. (a) the aligned source parse tree with target string; (b) general blocks alignment; (c) Tree-to-String rules, with root of VP. The tree structure is aligned with target strings

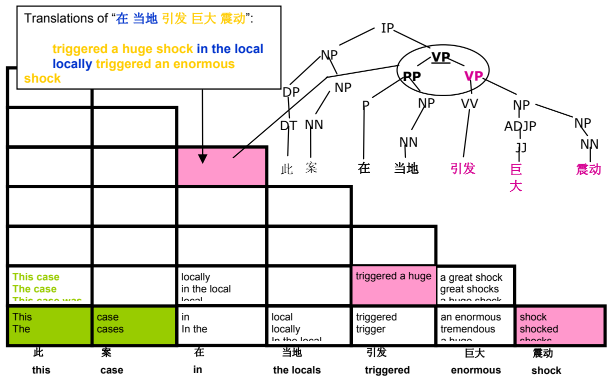


Figure 2: Subtree of "VP(PP,VP)" triggered a reordering pattern of swapping the order of the two children PP and VP in the source parse tree. This will move the translation "in the local" after the translation of "triggered a huge shock", to form the preferred translation in the highlighted cell: "triggered a huge shock in the local".

variable [X,0] and the label-based variable [X,VP] co-exist in our proposed rule.

We illustrate several special cases of our extracted Tree-to-String rules in the following. We index the variables with their positions to indicate the alignment $\sim$, and skip the feature function $\bar{w}$ to simplify the notations.

$$
\begin{aligned}
X \;\; \rightarrow< \;\; & [X, IP]; [X, NP0]\ [X, VP0]; \\
& [X, NP0]\ is\ [X, VP0] > .
\end{aligned} \tag{2}
$$

The rule in Eqn. 2 shows that a source tree rooted at IP, with two children of NP and VP generalized into variables [X,NP] and [X,VP]; they are rewritten into "[X,NP] is [X,VP]", with the spontaneous word *is* inserted. Such rules are not allowed in Hiero-style models, as there is no lexical item between the two variables (Chiang, 2005) in the source side. This rule will generate a spontaneous word "is" from the given subtree structure. Usually, it is very hard to align the spontaneous word correctly, and the rules we proposed indicate that spontaneous words are generated directly from the source sub-tree structure, and they might not necessarily get aligned to some particular source words.

A second example is shown in Eqn. 3, which is similar to the Hiero rules:

$$
\begin{aligned}
X \;\; \rightarrow< \;\; & \emptyset; [X, 0]\ \text{zhiyi}; \\
& one\ of\ the\ [X, 0] > .
\end{aligned} \tag{3}
$$

The rule in Eqn. 3 shows that when there is no linguistically-motivated root covering the span, ([X,NULL] is then assigned), we simply back off to the Hiero rules. In this case, the source span of $[X, 0]$ zhiyi is rewritten into the target "$one\ of\ the\ [X, 0]$", without considering the mapping of the root of the span. In this way, the representation power is kept in the variables in our rules, even if the source subtree is aligned to a discontinuous sequence on the target side. This is important for Chinese-to-English, because the grammar structure is so different that more than 40% of the subtree structures were not kept during the translation in our study on hand-aligned data. Following strictly the source side syntax will derail from these informative translation patterns.

$$X \rightarrow < [X, NP]; [X, NN1][X, NN2][X, NN3]; [X, NN3][X, NN1][X, NN2] > . \tag{4}$$

Eqn. 4. is a POS-based rule — a special case in our proposed rules. This rule shows the reordering patterns for three adjacent NN's. POS based rules can be very informative for some language-pairs such as Arabic-to-English, where the ADJ is usually moved before NN during the translations.

As also shown in Eqn. 4 for POS sequences, in the UPenn treebank-style parse trees, a root usually have more than two variables. Our rule set for subtree, therefore, contain more than two variables: "$X \rightarrow < [X, IP]; [X, ADVP0][X, NP0][X, VP0]; [X, NP0] [X, ADVP0][X, VP0] >$". A CYK-style decoder has to rely on *binarization* to preprocess the grammar as did in (Zhang et al., 2006) to handle multi-nonterminal rules. We adopt the so-called *dotted-rule* or *dotted-production*, similar to the Early-style algorithm (Earley, 1970), to handle the multi-nonterminal rules in our chart-based decoder.

3.2 Feature Functions

As used in most of the SMT decoders for a phrase-pair, a set of standard feature functions are applied in our decoder, including IBM Model-1 like scores in both directions, relative frequencies in both directions. In addition to these features, a counter is associated to each rule to collect how many rules were applied so far to generate a hypothesis. The standard Minimum Error Rate training (Och, 2003) was applied to tune the weights for all feature types.

The number of extracted rules from the GALE data is generally large. We pruned the rules according to their frequencies, and only keep at most the top-50 frequent candidates for each source side.

4 Chart-based Decoder

Given the source sentence, with constituent parse-trees, the decoder is to find the best derivation D^* which yield the English string e^*:

$$e^* = \arg\max_{D^*} \{\phi(D)\phi(e)\phi(f|e)\}, \tag{5}$$

where $\phi(D)$ is the cost for each of the derivations that lead to e from a given source-parsed f; $\phi(e)$ is for cost functions from the standard n-gram language models; $\phi(f|e)$ is the cost for the standard translation models, including general blocks. We separate the costs for normal blocks and the generalized rules explicitly here, because the blocks contain stronger lexical evidences observed directly from data, and we assign them with less cost penalties via a different weight factor visible for optimization, and prefer the lexical match over the derived paths during the decoding.

Our decoder is a chart-based parser with beam-search for each cell in a chart. Because the tree-structure can have more than two children, therefore, the Tree-to-String rules extracted usually contain more than two variables. Slightly different from the decoder in (Chiang, 2005), we implemented the *dotted-rule* in Early-style parser to handle rules containing more than two variables. Our cube-expansion, implemented the cube-pruning in Chiang (2007), and integrated *piece-wise* cost computations for language models via LM states. The intermediate hypotheses were merged (recombined) according to their LM states and other cost model states. We use MER (Och, 2003) to tune the decoder's parameters using a development data set.

Figure 2 shows an example of a tree-based rule fired at the subtree of VP covering the highlighted cell. When a rule is applied at a certain cell in the chart, the covered source ngram should match not only the lexical items in the rules, but also the tree-structures as well. The two children under the subtree root VP are PP ("在当地": in the local) and VP ("引发巨大震动": triggered a huge shock). This rule triggered a swap of these children to generate the correct word order in the translation: "triggered a huge shock in the local".

5 Experiments

Our training data consists of two corpora: the GALE Chinese-English parallel corpus and the LDC hand-aligned corpus[1]. The Chinese side of these two corpora were parsed using a constituency parser (Luo, 2003). The average labeled F-measure of the parser is 81.4%.

Parallel sentences were first word-aligned using a MaxEnt aligner (Ittycheriah and Roukos, 2005). Then, phrase-pairs that overlap with our development and test set were extracted from the word alignments (from both hand alignments and automatically aligned GALE corpora) based on the projection principle (Tillmann, 2003). Besides the regular phrase-pairs, we also extracted the Tree-to-String rules from the two corpora. The detailed statistics are shown in Table 1. Our re-implementation of Hiero system is the baseline. We integrated the eleven reordering rules described in (Wang et al., 2007), in our chart-based decoder. In addition, we report the results of using the Tree-to-String rules extracted from the hand-aligned training data and the automatically aligned training data. We also report the result of our translation quality in terms of both BLEU (Papineni et al., 2002) and TER (Snover et al., 2006) against four human reference translations.

5.1 The Data

Table 1 shows the statistics of our training, development and test data. As our word aligner (Ittycheriah and Roukos, 2005) can introduce errors in extracting Tree-to-String rules, we use a small hand-aligned data set "CE16K", which consists of 16K sentence-pairs, to get relatively clean rules, free from alignment errors. A much larger GALE data set, which consists of 10 million sentence-pairs, is used to investigate the scalability of our proposed approach.

Table 1: Training and Test Data

Train/test	sentences	src words	tgt words
CE16K	16379	380103	477801
GALE	10.5M	274M	310M
MT03	919	24099	-
Dev07	2303	61881	-

[1]LDC2006E93

The NIST 2003 MT Evaluation (MT03) is used as our development data set to tune the decoder's parameters toward better BLEU score. The text part of GALE 2007 Chinese-to-English Development set (GALE DEV07) is used as our test set. MT03 consists of 919 sentences, whereas GALE DEV07 consists of 2303 sentences under two genres: NewsWire and WebLog. Both have four human reference translations.

5.2 Details of Extracted Rules

From the hand-aligned data, the rules we extracted fall into three categories: regular blocks (phrase-pairs), Hiero-like rules, and Tree-to-String rules. The statistics of the extracted rules are shown in Table 2

Table 2: Rules extracted from hand-aligned data

Types	Frequency
Block	846965
Hiero	508999
Tree-to-String	409767
Total	1765731

We focus on Tree-to-String rules. Table 3 shows the detailed statistics of the Tree-to-String rules extracted from the Chinese-to-English hand-aligned training data. The following section provides a detailed analysis of the most frequent subtrees observed in our training data.

5.2.1 Frequent Subtrees: NP, VP, and DNP

The majority of Tree-to-String rules we extracted are rooted at the following labels: NP (46%), VP(22.8%), DNP (2.23%), and QP(2.94%).

Wang et al. (2007) covers only subtrees of NP, VP, and LCP, which are a subset of our proposed Tree-to-String rules here. They apply these rules as a pre-processing step to reorder the input sentences with hard decisions. Our proposed Tree-to-String rules, on the contrary, are applied during the decoding process which allows for considering many possible competing reordering options for the given sentences, and the decoder will choose the best one according to the cost functions.

Table 4 shows the statistics of reordering rules for subtrees rooted at VP. The statistics suggest that

Table 5: Hiero, Tree-Based (eleven rules in Wang et al. (2007)), and Tree-to-String Rules with "DE"

Ruleset	Root	Src	Tgt	Frequency
Hiero	NULL	[X,0] 的 [X,1]	[X,0] 's [X,1]	347
	NULL	[X,0] 的 [X,1]	[X,1] of [X,0]	306
	NULL	[X,0] 的 [X,1]	[X,0] of [X,1]	174
Tree-Based	NP	DNP(NP) NP	NP DNP(NP)	-
	NP	DNP(PP) NP	NP DNP(PP)	-
	NP	DNP(LCP) NP	NP DNP(LCP)	-
Tree-to-String	[X,DNP]	[X,NP] [X,DEG]	[X,NP] [X,DEG]	580
	[X,DNP]	[X,NP] [X,DEG]	[X,DEG] [X,NP]	2163
	[X,DNP]	[X,NP] [X,DEG]	[X,NP] , [X,DEG]	4

Table 3: Distributions of the NP, VP, QP, LCP rules

Root	Frequency	Percentage (%)
NP	189616	46.2
VP	93535	22.8
IP	68341	16.6
PP	18519	4.51
DNP	9141	2.23
QP	12064	2.94
LCP	4127	1.00
CP	2994	0.73
PRN	2810	0.68
DP	1415	0.34
Others	6879	1.67
Total	409767	-

Table 4: Distribution of the reordering rules for subtrees rooted at VP: [X,VP]; [X,PP] [X,VP]; statistics are collected from GALE training data

Root	Target	Frequency
VP	[X,PP] [X,VP]	126310
	[X,VP] [X,PP]	22144
	[X,PP] , [X,VP]	1524
	[X,PP] that [X,VP]	1098
	[X,PP] and [X,VP]	831

it is impossible to come up with a reordering rule that is always applicable. For instance, (Wang et al., 2007) will always swap the children of the subtree VP(PP,VP). However, the statistics shown in Table 4 suggest that might not be best way. In fact, due to parser's performance and word alignment accuracies, the statistics we collected from the GALE dataset, containing 10 million sentence-pairs, show that the children in the subtree VP(PP,VP) is translated monotonically 126310 times, while reordered of only 22144 times. However, the hand-aligned data support the swap for 1245 times, and monotonically for only 168 times. Part of this disagreement is due to the word segmentation errors, incorrect word alignments and unreliable parsing results.

Another observations through our extracted Tree-to-String rules is on the controlled insertion of the target spontaneous[2] (function) words. Instead of hypothesizing spontaneous words based only on the language model or only on observing in phrase-pairs, we make use of the Tree-to-String rules to get suggestion on the insertion of spontaneous words. In this way, we can make sure that the spontaneous words are generated from the structure information, as opposed to those from a pure hypothesis. The advantage of this method is shown in Table 4. For instance, the word "that" and the punctuation "," were generated in the target side of the rule. This proves that our model can provide a more principled way to generate spontaneous words needed for fluent translations.

5.2.2 DEG and DEC

An interesting linguistic phenomenon that we investigated is the Chinese word DE "的". "的" is an informative lexical clue that indicates the need for long range phrasal movements. Table 5 shows a few

[2]Target spontaneous words are function words that do not have specific lexical source informants and are needed to make the target translation fluent.

high-frequent reordering rules that contain the Chinese word "DE".

The three type of rules handle "DE" differently. A major difference is the structure in the source side. Hiero rules do not consider any structure, and apply the rule of "[X,0] 的 [X,1]". Tree-based rules, as described in Wang et al. (2007) do not handle 的 directly; they are often implicitly taken care of when reordering DNPs instead. Our proposed Tree-to-String rules model 的 directly in a subtree containing DEG/DEC, which triggers word reordering within the structure. Our rule set includes all the above three rule-types with the associated frequencies, this enriched the reordering choices to be chosen by the chart-based decoder, guided by the statistics collected from the data and the language model costs.

5.3 Evaluation

We tuned the decoding parameters using the MT03 data set, and applied the updated parameters to the GALE evaluation set. The *eleven* rules of VP, NP, and LCP (tree-based) improved the Hiero baseline[3] from 32.43 to 33.02 on BLEU. The reason, the tree-reordering does not gain much over Hiero baseline, is probably that the reordering patterns covered by tree-reordering rules, are potentially handled in the standard Hiero grammar.

A small but noticeable further improvement over tree-based rules, from 33.02 to 33.26, was obtained on applying Tree-to-String rules extracted from hand-aligned dataset. We think that the Tree-based rules covers major reordering patterns for Chinese-English, and our hand-aligned dataset is also too small to capture representative statistics and more reordering patterns. A close check at the rules we learned from the hand-aligned data shows that the tree-based rules are simply the subset of the rules extracted. The Tree-to-String grammar improved the Hiero baseline from 32.43 to 33.26 on BLEU; considering the effects from the tree-based rules only, the additional information improved the BLEU scores from 33.02 to 33.26. Similar pictures of improvements were observed for the two unseen tests of newswire and weblog in GALE data.

When applying the rules extracted from the much larger GALE training set with about ten million sentence-pairs, we achieved significant improvements from both genres (newswire and web data). The improvements are significant in both BLEU and TER. BLEU improved from 32.44 to 33.51 on newswire, and from 25.88 to 27.91 on web data. Similar improvements were found in TER as shown in the table. The gain came mostly from the richer extracted rule set, which not only presents robust statistics for reordering patterns, but also offers more target spontaneous words generated from the syntactic structures. Since the top-frequent rules extracted are NP, VP, and IP as shown in Table 3, our proposed rules will be able to win the correct word order with reliable statistics, as long as the parser shows acceptable performances on these structures. This is especially important for weblog data, where the parser's overall accuracy potentially might not be very good.

Table 7 shows the translations from different grammars for the same source sentence. Both Tree-based and Tree-to-String methods get the correct reordering, while the latter can suggest insertions of target spontaneous words like "a" to allow the translation to run more fluently.

6 Conclusion and Discussions

In this paper, we proposed our approach to model both local and non-local word-reordering in one probabilistic synchronous CFG. Our current model incorporates source-side syntactic information, to model the observations that the source syntactic constituent tends to move together during translations. The proposed rule set generalizes over the variables in Hiero-rules, and we also showed the special cases of the Tree-based rules and the POS-based rules. Since the proposed rules has at most one-level tree structure, they can be easily applied in a chart-based decoder. We analyzed the statistics of our rules, qualitatively and quantitatively. Next, we compared our work with other research, especially with the work in Wang et al. (2007). Finally, we reported our empirical results on Chinese-English translations. Our Tree-to-String rules showed significant improvements over the Hiero baseline on the GALE DEV07 test set.

Given the low accuracy of the parsers, and the potential errors from Chinese word-segmentations, and

[3]Hiero results are from our own re-implementation.

Table 6: Hiero, Tree-Based (NP, VP, LCP), and Tree-to-String rules extracted from hand-aligned data (H) or from GALE training data (G)

Setup	MT03		GALE07-NewsWire		GALE07-Weblog	
	BLEUr4n4	TER	BLEUr4n4	TER	BLEUr4n4	TER
Hiero	32.43	59.75	31.68	61.45	25.99	65.65
Tree-based	33.02	59.84	32.22	61.46	25.67	65.64
Tree-to-String (H)	33.26	61.04	32.44	61.36	25.88	65.54
Tree-to-String (G)	35.51	57.28	33.51	59.71	27.91	62.88

Table 7: Hiero, Tree-Based (NP, VP, LCP), Tree-to-String Translations

Src-Sent	此案在当地引发巨大震动。
Hiero	in this case local triggered shock .
Tree-Based	the case triggered uproar in the local.
Tree-to-String	the case triggered a huge uproar in the local .

word-alignments, our rules learned are still noisy. Exploring better cost functions associate each rule might lead to further improvement. Because of the relative high accuracy of English parsers, many works such as Zollmann and Venugopal (2006) and Shen et al. (2008) emphasize on using syntax in target languages, to directly influence the fluency aspect of the translation output. In future, we plan to incorporate features from target-side syntactic information, and connect them with the source information explored in this paper, to model long-distance reordering for better translation quality.

Acknowledgments

The authors would like to thank the anonymous reviewers for their comments to improve this paper. This work was supported by DARPA GALE program under the contract number HR0011-06-2-0001.

References

Yaser Al-Onaizan and Papineni. Kishore. 2006. Distortion models for statistical machine translation. In *Proceedings of ACL-COLING*, pages 529–536.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270, Ann Arbor, Michigan, June. Association for Computational Linguistics.

David Chiang. 2007. Hierarchical phrase-based translation. In *Computational Linguistics*.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL*.

Jay Earley. 1970. An efficient context-free parsing algorithm. In *Communications of the ACM.*, volume 13, pages 94–102.

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.

Jonathan Graehl and Kevin Knight. 2004. Training tree transducers. In *Proc. NAACL-HLT*.

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

Shankar Kumar and William Byrne. 2005. Local phrase reordering models for statistical machine translation. In *HLT/EMNLP 2005*, Vancouver, B.C., Canada.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *ACL-Coling*.

Yang Liu, Yun Huang, Qun Liu, and Shouxun Lin. 2007. Forest-to-string statistical translation rules. In *45th Annual Meeting of the Association for Computational Linguistics*.

Xiaoqiang Luo. 2003. A maximum entropy chinese character-based parser. In *Proc. of ACL*.

Yuval Marton and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrased-based translation. In *ACL*.

Sonja Niessen and Hermann Ney. 2004. Statistical machine translation with scarce resources using morphosyntactic information. In *Computational Linguistics*.

Franz J. Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. In *Computational Linguistics*, volume 30, pages 417–449.

Franz J. Och. 2003. Minimum error rate training for statistical machine translation. In *Proc. of the 41st Annual Meeting of the Association for Computational Linguistics*, Japan, Sapporo, July.

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.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL*.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *AMTA*.

Christoph Tillmann and Tong Zhang. 2005. A localized prediction model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 557–564, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Christoph Tillmann. 2003. A projection extension algorithm for statistical machine translation. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*.

Christoph Tillmann. 2008. A rule-driven dynamic programming decoder for statistical mt. In *HLT Second Workshop on Syntax and Structure in Statistical Translation*.

Chao Wang, Michael Collins, and Phillip Koehn. 2007. Chinese syntactic reordering for statistical machine translation. In *proceedings of EMNLP*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. In *Computational Linguistics*, volume 23(3), pages 377–403.

Fei Xia and Michael McCord. 2004. Improving a statistical mt system with automatically learned rewrite patterns. In *the 20th International Conference on Computational Linguistics (COLING 2004)*, Geneva, Switzerland, Aug 22-29.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *ACL-Coling*.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. In *Natural Language Engineering*, volume 11, pages 207–238.

K. Yamada and Kevin. Knight. 2001. Syntax-based Statistical Translation Model. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL-2001)*.

Richard Zens, E. Matusov, and Hermmann Ney. 2004. Improved word alignment using a symmetric lexicon model. In *Proceedings of the 20th International Conference on Computational Linguistics (CoLing 2004)*, pages 36–42, Geneva, Switzerland, Auguest.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proceedings of the HLT-NAACL*.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proc. of NAACL 2006 - Workshop on statistical machine translation*.

Weakly-Supervised Acquisition of Labeled Class Instances using Graph Random Walks

Partha Pratim Talukdar[*]
University of Pennsylvania
Philadelphia, PA 19104
partha@cis.upenn.edu

Joseph Reisinger[*]
University of Texas at Austin
Austin, TX 78712
joeraii@cs.utexas.edu

Marius Paşca
Google Inc.
Mountain View, CA 94043
mars@google.com

Deepak Ravichandran
Google Inc.
Mountain View, CA 94043
deepakr@google.com

Rahul Bhagat[*]
USC Information Sciences Institute
Marina Del Rey, CA 90292
rahul@isi.edu

Fernando Pereira
Google Inc.
Mountain View, CA 94043
pereira@google.com

Abstract

We present a graph-based semi-supervised label propagation algorithm for acquiring open-domain labeled classes and their instances from a combination of unstructured and structured text sources. This acquisition method significantly improves coverage compared to a previous set of labeled classes and instances derived from free text, while achieving comparable precision.

1 Introduction

1.1 Motivation

Users of large document collections can readily acquire information about the instances, classes, and relationships described in the documents. Such relations play an important role in both natural language understanding and Web search, as illustrated by their prominence in both Web documents and among the search queries submitted most frequently by Web users (Jansen et al., 2000). These observations motivate our work on algorithms to extract instance-class information from Web documents.

While work on named-entity recognition traditionally focuses on the acquisition and identification of instances within a small set of coarse-grained classes, the distribution of instances within query logs indicates that Web search users are interested in a wider range of more fine-grained classes. Depending on prior knowledge, personal interests and immediate needs, users submit for example medical queries about the symptoms of *leptospirosis* or the treatment of *monkeypox*, both of which are instances of *zoonotic diseases*, or the risks and benefits of *surgical procedures* such as *PRK* and *angioplasty*. Other users may be more interested in *African countries* such as *Uganda* and *Angola*, or *active volcanoes* like *Etna* and *Kilauea*. Note that *zoonotic diseases*, *surgical procedures*, *African countries* and *active volcanoes* serve as useful class labels that capture the semantics of the associated sets of class instances. Such interest in a wide variety of specific domains highlights the utility of constructing large collections of fine-grained classes.

Comprehensive and accurate class-instance information is useful not only in search but also in a variety of other text processing tasks including co-reference resolution (McCarthy and Lehnert, 1995), named entity recognition (Stevenson and Gaizauskas, 2000) and seed-based information extraction (Riloff and Jones, 1999).

1.2 Contributions

We study the acquisition of open-domain, labeled classes and their instances from both structured and unstructured textual data sources by combining and ranking individual extractions in a principled way with the Adsorption label-propagation algorithm (Baluja et al., 2008), reviewed in Section 3 below.

A collection of labeled classes acquired from text (Van Durme and Paşca, 2008) is extended in two ways:

1. Class label coverage is increased by identifying additional class labels (such as *public agencies* and *governmental agencies*) for existing

[*]Contributions made during internships at Google.

instances such as *Office of War Information*),

2. The overall instance coverage is increased by extracting additional instances (such as *Addison Wesley* and *Zebra Books*) for existing class labels (*book publishers*).

The WebTables database constructed by Cafarella et al. (2008) is used as the source of additional instances. Evaluations on gold-standard labeled classes and instances from existing linguistic resources (Fellbaum, 1998) indicate coverage improvements relative to that of Van Durme and Paşca (2008), while retaining similar precision levels.

2 First Phase Extractors

To show Adsorption's ability to uniformly combine extractions from multiple sources and methods, we apply it to: 1) high-precision open-domain extractions from free Web text (Van Durme and Paşca, 2008), and 2) high-recall extractions from WebTables, a large database of HTML tables mined from the Web (Cafarella et al., 2008). These two methods were chosen to be representative of two broad classes of extraction sources: free text and structured Web documents.

2.1 Extraction from Free Text

Van Durme and Paşca (2008) produce an open-domain set of instance clusters $C \in \mathcal{C}$ that partitions a given set of instances $\mathcal{I}$ using distributional similarity (Lin and Pantel, 2002), and labels using *is-a* patterns (Hearst, 1992). By filtering the class labels using distributional similarity, a large number of high-precision labeled clusters are extracted. The algorithm proceeds iteratively: at each step, all clusters are tested for label *coherence* and all coherent labels are tested for high cluster *specificity*. Label L is coherent if it is shared by at least $J\%$ of the instances in cluster C, and it is specific if the total number of other clusters $C' \in \mathcal{C}, C' \neq C$ containing instances with label L is less than K. When a cluster is found to match these criteria, it is removed from $\mathcal{C}$ and added to an output set. The procedure terminates when no new clusters can be removed from $\mathcal{C}$. Table 1 shows a few randomly chosen classes and representative instances obtained by this procedure.

2.2 Extraction from Structured Text

To expand the instance sets extracted from free text, we use a *table-based extraction* method that mines structured Web data in the form of HTML tables. A significant fraction of the HTML tables in Web pages is assumed to contain coherent lists of instances suitable for extraction. Identifying such tables from scratch is hard, but seed instance lists can be used to identify potentially coherent table columns. In this paper we use the WebTables database of around 154 million tables as our structured data source (Cafarella et al., 2008).

We employ a simple ranking scheme for candidate instances in the WebTables corpus $\mathcal{T}$. Each table $\mathbf{T} \in \mathcal{T}$ consists of one or more columns. Each column $g \in \mathbf{T}$ consists of a set of candidate instances $i \in g$ corresponding to row elements. We define the set of unique *seed matches* in g relative to semantic class $C \in \mathcal{C}$ as

$$M_C(g) \overset{\text{def}}{=} \{i \in I(C) : i \in g\}$$

where $I(C)$ denotes the set of instances in seed class C. For each column g, we define its α-*unique class coverage*, that is, the set of classes that have at least α unique seeds in g,

$$Q(g;\alpha) \overset{\text{def}}{=} \{C \in \mathcal{C} : |M_C(g)| \geq \alpha\}.$$

Using M and Q we define a method for scoring columns relative to each class. Intuitively, such a score should take into account not only the number of matches from class C, but also the total number of classes that contribute to Q and their relative overlap. Towards this end, we introduce the scoring function

$$score(C, g; \alpha) \overset{\text{def}}{=} \underbrace{|M_C(g)|}_{\text{seed matches}} \cdot \overbrace{\frac{|M_C(g)|}{|\bigcup_{C' \in Q(g;\alpha)} I(C')|}}^{\text{class coherence}}$$

which is the simplest scoring function combining the number of seed matches with the coherence of the table column. Coherence is a critical notion in WebTables extraction, as some tables contain instances across many diverse seed classes, contributing to extraction noise. The class coherence introduced here also takes into account class overlap; that

Class	Size	Examples of Instances
Book Publishers	70	crown publishing, kluwer academic, prentice hall, puffin
Federal Agencies	161	catsa, dhs, dod, ex-im bank, fsis, iema, mema, nipc, nmfs, tdh, usdot
Mammals	956	armadillo, elephant shrews, long-tailed weasel, river otter, weddell seals, wild goat
NFL Players	180	aikman, deion sanders, fred taylor, jamal lewis, raghib ismail, troy vincent
Scientific Journals	265	biometrika, european economic review, nature genetics, neuroscience
Social Issues	210	gender inequality, lack of education, substandard housing, welfare dependency
Writers	5089	bronte sisters, hemingway, kipling, proust, torquato tasso, ungaretti, yeats

Table 1: A sample of the open-domain classes and associated instances from (Van Durme and Paşca, 2008).

is, a column containing many semantically similar classes is penalized less than one containing diverse classes.[1] Finally, an extracted instance i is assigned a score relative to class C equal to the sum of all its column scores,

$$score(i, C; \alpha) \overset{\text{def}}{=} \frac{1}{Z_C} \sum_{g \in \mathbf{T}, \mathbf{T} \in \mathcal{T}} score(C, g; \alpha)$$

where Z_C is a normalizing constant set to the maximum score of any instance in class C. This scoring function assigns high rank to instances that occur frequently in columns with many seed matches and high class specificity.

The ranked list of extracted instances is post-filtered by removing all instances that occur in less than d unique Internet domains.

3 Graph-Based Extraction

To combine the extractions from both free and structured text, we need a representation capable of encoding efficiently all the available information. We chose a graph representation for the following reasons:

- Graphs can represent complicated relationships between classes and instances. For example, an ambiguous instance such as *Michael Jordan* could belong to the class of both *Professors* and *NBA players*. Similarly, an instance may belong to multiple nodes in the hierarchy of classes. For example, *Blue Whales* could belong to both classes *Vertebrates* and *Mammals*, because *Mammals* are a subset of *Vertebrates*.
- Extractions from multiple sources, such as Web queries, Web tables, and text patterns can be represented in a single graph.
- Graphs make explicit the potential paths of information propagation that are implicit in the more common local heuristics used for weakly-supervised information extraction. For example, if we know that the instance *Bill Clinton* belongs to both classes *President* and *Politician* then this should be treated as evidence that the class of *President* and *Politician* are related.

Each instance-class pair (i, C) extracted in the first phase (Section 2) is represented as a weighted edge in a graph $G = (V, E, W)$, where V is the set of nodes, E is the set of edges and $W : E \rightarrow \mathbb{R}^+$ is the weight function which assigns positive weight to each edge. In particular, for each (i, C, w) triple from the set of base extractions, i and C are added to V and (i, C) is added to E, [2] with $W(i, C) = w$. The weight w represents the total score of all extractions with that instance and class. Figure 1 illustrates a portion of a sample graph. This simple graph representation could be refined with additional types of nodes and edges, as we discuss in Section 7.

In what follows, all nodes are treated in the same way, regardless of whether they represent instances or classes. In particular, all nodes can be assigned class labels. For an instance node, that means that the instance is hypothesized to belong to the class; for a class node, that means that the node's class is hypothesized to be semantically similar to the label's class (Section 5).

We now formulate the task of assigning labels to nodes as graph label propagation. We are given a

[1]Note that this scoring function does not take into account class containment: if all seeds are both *wind Instruments* and *instruments*, then the column should assign higher score to the more specific class.

[2]In practice, we use two directed edges, from i to C and from C to i, both with weight w.

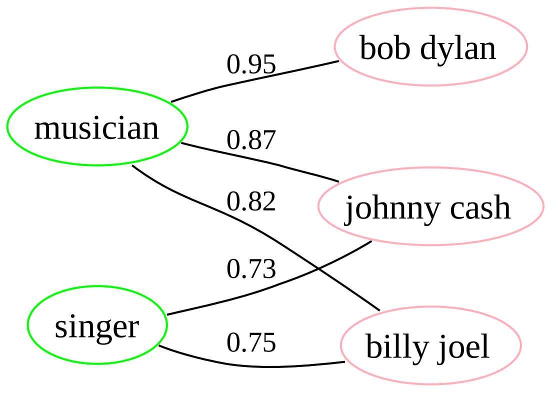


Figure 1: Section of a graph used as input into Adsorption. Though the nodes do not have any type associated with them, for readability, instance nodes are marked in pink while class nodes are shown in green.

set of instances $\mathcal{I}$ and a set of classes $\mathcal{C}$ represented as nodes in the graph, with connecting edges as described above. We annotate a few instance nodes with labels drawn from $\mathcal{C}$. That is, classes are used both as nodes in the graph and as labels for nodes. There is no necessary alignment between a class node and any of the (class) labels, as the final labels will be assigned by the Adsorption algorithm.

The Adsorption label propagation algorithm (Baluja et al., 2008) is now applied to the given graph. Adsorption is a general framework for label propagation, consisting of a few nodes annotated with labels and a rich graph structure containing the universe of all labeled and unlabeled nodes. Adsorption proceeds to label all nodes based on the graph structure, ultimately producing a probability distribution over labels for each node.

More specifically, Adsorption works on a graph $G = (V, E, W)$ and computes for each node v a *label distribution* L_v that represents which labels are more or less appropriate for that node. Several interpretations of Adsorption-type algorithms have appeared in various fields (Azran, 2007; Zhu et al., 2003; Szummer and Jaakkola, 2002; Indyk and Matousek, 2004). For details, the reader is referred to (Baluja et al., 2008). We use two interpretations here:

Adsorption through Random Walks: Let $G_r = (V, E_r, W_r)$ be the edge-reversed version of the original graph $G = (V, E, W)$ where $(a, b) \in E_r$ iff $(b, a) \in E$; and $W_r(a, b) = W(b, a)$. Now, choose a node of interest $q \in V$. To estimate L_q for q, we perform a random walk on G_r starting from q to generate values for a random label variable L. After reaching a node v during the walk, we have three choices:

1. With probability p_v^{cont}, continue the random walk to a neighbor of v.
2. With probability p_v^{abnd}, abandon the random walk. This abandonment probability makes the random walk stay relatively close to its source when the graph has high-degree nodes. When the random walk passes through such a node, it is likely that further transitions will be into regions of the graph unrelated to the source. The abandonment probability mitigates that effect.
3. With probability p_v^{inj}, stop the random walk and emit a label L from I_v.

L_q is set to the expectation of all labels L emitted from random walks initiated from node q.

Adsorption through Averaging: For this interpretation we make some changes to the original graph structure and label set. We extend the label distributions L_v to assign a probability not only to each label in $\mathcal{C}$ but also to the *dummy* label $\perp$, which represents lack of information about the actual label(s). We represent the initial knowledge we have about some node labels in an *augmented* graph $G' = (V', E', W')$ as follows. For each $v \in V$, we define an *initial* distribution $I_v = L^{\perp}$, where $L^{\perp}$ is the *dummy* distribution with $L^{\perp}(\perp) = 1$, representing lack of label information for v. In addition, let $V_s \subseteq V$ be the set of nodes for which we have some actual label knowledge, and let $V' = V \cup \{\bar{v} : v \in V_s\}, E' = E \cup \{(\bar{v}, v) : v \in V_s\}$, and $W'(\bar{v}, v) = 1$ for $v \in V_s$, $W'(u, v) = W(u, v)$ for $u, v \in V$. Finally, let $I_{\bar{v}}$ (seed labels) specify the knowledge about possible labels for $v \in V_s$. Less formally, the $\bar{v}$ nodes in G' serve to *inject* into the graph the prior label distributions for each $v \in V_s$.

The algorithm proceeds as follows: For each node use a fixed-point computation to find label distributions that are weighted averages of the label distributions for all their neighbors. This causes the non-dummy initial distribution of V_s nodes to be propagated across the graph.

Baluja et al. (2008) show that those two views are equivalent. Algorithm 1 combines the two views: instead of a random walk, for each node v, it iteratively computes the weighted average of label distributions from neighboring nodes, and then uses the random walk probabilities to estimate a new label distribution for v.

For the experiments reported in Section 4, we used the following heuristics from Baluja et al. (2008) to set the random walk probabilities:

- Let $c_v = \frac{\log \beta}{\log(\beta + \exp H(v))}$ where $H(v) = -\sum_u p_{uv} \times \log(p_{uv})$ with $p_{uv} = \frac{W(u,v)}{\sum_{u'} W(u',v)}$. $H(v)$ can be interpreted as the entropy of v's neighborhood. Thus, c_v is lower if v has many neighbors. We set $\beta = 2$.

- $j_v = (1 - c_v) \times \sqrt{H(v)}$ if $I_v \neq L^\top$ and 0 otherwise.

- Then let

$$\begin{aligned} z_v &= \max(c_v + j_v, 1) \\ p_v^{cont} &= c_v / z_v \\ p_v^{inj} &= j_v / z_v \\ p_v^{abnd} &= 1 - p_v^{cont} - p_v^{abnd} \end{aligned}$$

  Thus, abandonment occurs only when the continuation and injection probabilities are low enough.

The algorithm is run until convergence which is achieved when the label distribution on each node ceases to change within some tolerance value. Alternatively, the algorithm can be run for a fixed number of iterations which is what we used in practice[3].

Finally, since Adsorption is memoryless, it easily scales to tens of millions of nodes with dense edges and can be easily parallelized, as described by Baluja et al. (2008).

[3]The number of iterations was set to 10 in the experiments reported in this paper.

Algorithm 1 Adsorption Algorithm.
Input: $G' = (V', E', W')$, I_v $(\forall v \in V')$.
Output: Distributions $\{L_v : v \in V\}$.

```
1: $L_v = I_v \; \forall v \in V'$
2:
3: **repeat**
4:     $N_v = \sum_u W(u,v)$
5:     $D_v = \frac{1}{N_v} \sum_u W(u,v) L_u \;\; \forall v \in V'$
6:     **for all** $v \in V'$ **do**
7:         $L_v = p_v^{cont} \times D_v + p_v^{inj} \times I_v + p_v^{abnd} \times L^\top$
8:     **end for**
9: **until** convergence
```

4 Experiments

4.1 Data

As mentioned in Section 3, one of the benefits of using Adsorption is that we can combine extractions by different methods from diverse sources into a single framework. To demonstrate this capability, we combine extractions from free-text patterns and from Web tables. To the best of our knowledge, this is one of the first attempts in the area of minimally-supervised extraction algorithms where unstructured and structured text are used in a principled way within a single system.

Open-domain (instance, class) pairs were extracted by applying the method described by Van Durme and Paşca (2008) on a corpus of over 100M English web documents. A total of 924K (instance, class) pairs were extracted, containing 263K unique instances in 9081 classes. We refer to this dataset as A8.

Using A8, an additional 74M unique (instance,class) pairs are extracted from a random 10% of the WebTables data, using the method outlined in Section 2.2. For maximum coverage we set $\alpha = 2$ and $d = 2$, resulting in a large, but somewhat noisy collection. We refer to this data set as WT.

4.2 Graph Creation

We applied the graph construction scheme described in Section 3 on the A8 and WT data combined, resulting in a graph with 1.4M nodes and 75M edges. Since extractions in A8 are not scored, weight of all

Seed Class	Seed Instances
Book Publishers	millbrook press, academic press, springer verlag, chronicle books, shambhala publications
Federal Agencies	dod, nsf, office of war information, tsa, fema
Mammals	african wild dog, hyaena, hippopotamus, sperm whale, tiger
NFL Players	ike hilliard, isaac bruce, torry holt, jon kitna, jamal lewis
Scientific Journals	american journal of roentgenology, pnas, journal of bacteriology, american economic review, ibm systems journal

Table 2: Classes and seeds used to initialize Adsorption.

edges originating from A8 were set at 1[4]. This graph is used in all subsequent experiments.

5 Evaluation

We evaluated the Adsorption algorithm under two experimental settings. First, we evaluate Adsorption's extraction precision on (instance, class) pairs obtained by Adsorption but not present in A8 (Section 5.1). This measures whether Adsorption can add to the A8 extractions at fairly high precision. Second, we measured Adsorption's ability to assign labels to a fixed set of gold instances drawn from various classes (Section 5.2).

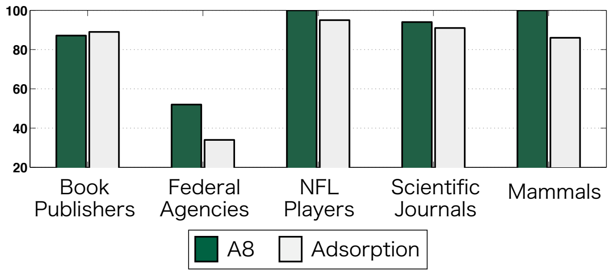


Figure 2: Precision at 100 comparisons for A8 and Adsorption.

5.1 Instance Precision

First we manually evaluated precision across five randomly selected classes from A8: Book Publishers, Federal Agencies, NFL Players, Scientific Journals and Mammals. For each class, 5 seed instances were chosen manually to initialize Adsorption. These classes and seeds are shown in Table 2. Adsorption was run for each class separately and the resulting ranked extractions were manually evaluated.

Since the A8 system does not produce ranked lists of instances, we chose 100 random instances from the A8 results to compare to the top 100 instances produced by Adsorption. Each of the resulting 500 instance-class pairs (i, C) was presented to two human evaluators, who were asked to evaluate whether the relation "i is a C" was correct or incorrect. The user was also presented with Web search link to verify the results against actual documents. Results from these experiments are presented in Figure 2 and Table 4. The results in Figure 2 show that the A8 system has higher precision than the Adsorption system. This is not surprising since the A8 system is tuned for high precision. When considering individual evaluation classes, changes in precision scores between the A8 system and the Adsorption system vary from a small increase from 87% to 89% for the class Book Publishers, to a significant decrease from 52% to 34% for the class Federal Agencies, with a decrease of 10% as an average over the 5 evaluation classes.

Class	Precision at 100 (non-A8 extractions)
Book Publishers	87.36
Federal Agencies	29.89
NFL Players	94.95
Scientific Journals	90.82
Mammal Species	84.27

Table 4: Precision of top 100 Adsorption extractions (for five classes) which were **not** present in A8.

Table 4 shows the precision of the Adsorption system for instances not extracted by the A8 system.

[4]A8 extractions are assumed to be high-precision and hence we assign them the highest possible weight.

Seed Class	Non-Seed Class Labels Discovered by Adsorption
Book Publishers	small presses, journal publishers, educational publishers, academic publishers, commercial publishers
Federal Agencies	public agencies, governmental agencies, modulation schemes, private sources, technical societies
NFL Players	sports figures, football greats, football players, backs, quarterbacks
Scientific Journals	prestigious journals, peer-reviewed journals, refereed journals, scholarly journals, academic journals
Mammal Species	marine mammal species, whale species, larger mammals, common animals, sea mammals

Table 3: Top class labels ranked by their similarity to a given seed class in Adsorption.

Seed Class	Sample of Top Ranked Instances Discovered by Adsorption
Book Publishers	small night shade books, house of anansi press, highwater books, distributed art publishers, copper canyon press
NFL Players	tony gonzales, thabiti davis, taylor stubblefield, ron dixon, rodney hannah
Scientific Journals	journal of physics, nature structural and molecular biology, sciences sociales et santé, kidney and blood pressure research, american journal of physiology–cell physiology

Table 5: Random examples of top ranked extractions (for three classes) found by Adsorption which were not present in A8.

Such an evaluation is important as one of the main motivations of the current work is to increase coverage (recall) of existing high-precision extractors without significantly affecting precision. Results in Table 4 show that Adsorption is indeed able to extraction with high precision (in 4 out of 5 cases) new instance-class pairs which were not extracted by the original high-precision extraction set (in this case A8). Examples of a few such pairs are shown in Table 5. This is promising as almost all state-of-the-art extraction methods are high-precision and low-recall. The proposed method shows a way to overcome that limitation.

As noted in Section 3, Adsorption ignores node type and hence the final ranked extraction may also contain classes along with instances. Thus, in addition to finding new instances for classes, it also finds additional class labels similar to the seed class labels with which Adsorption was run, at no extra cost. Some of the top ranked class labels extracted by Adsorption for the corresponding seed class labels are shown in Table 3. To the best of our knowledge, there are no other systems which perform both tasks simultaneously.

5.2 Class Label Recall

Next we evaluated each extraction method on its relative ability to assign labels to class instances. For each test instance, the five most probably class labels are collected using each method and the Mean Reciprocal Rank (MRR) is computed relative to a gold standard target set. This target set, WN-gold, consists of the 38 classes in Wordnet containing 100 or more instances.

In order to extract meaningful output from Adsorption, it is provided with a number of labeled seed instances (1, 5, 10 or 25) from each of the 38 test classes. Regardless of the actual number of seeds used as input, all 25 seed instances from each class are removed from the output set from all methods, in order to ensure fair comparison.

The results from this evaluation are summarized in Table 6; AD x refers to the adsorption run with x seed instances. Overall, Adsorption exhibits higher MRR than either of the baseline methods, with MRR increasing as the amount of supervision is increased. Due to its high coverage, WT assigns labels to a larger number of the instance in WN-gold than any other method. However, the average rank of the correct class assignment is lower, resulting is

Method	MRR (full)	MRR (found only)	# found
A8	0.16	0.47	2718
WT	0.15	0.21	**5747**
AD 1	0.26	0.45	4687
AD 5	0.29	0.48	4687
AD 10	0.30	0.51	4687
AD 25	**0.32**	**0.55**	4687

Table 6: Mean-Reciprocal Rank scores of instance class labels over 38 Wordnet classes (WN-gold). MRR (full) refers to evaluation across the entire gold instance set. MRR (found only) computes MRR only on recalled instances.

lower MRR scores compared to Adsorption. This result highlights Adsorption's ability to effectively combine high-precision, low-recall (A8) extractions with low-precision, high-recall extractions (WT) in a manner that improves *both* precision and coverage.

6 Related Work

Graph based algorithms for minimally supervised information extraction methods have recently been proposed. For example, Wang and Cohen (2007) use a random walk on a graph built from entities and relations extracted from semi-structured text. Our work differs both conceptually, in terms of its focus on open-domain extraction, as well as methodologically, as we incorporate both unstructured and structured text. The re-ranking algorithm of Bellare et al. (2007) also constructs a graph whose nodes are instances and attributes, as opposed to instances and classes here. Adsorption can be seen as a generalization of the method proposed in that paper.

7 Conclusion

The field of open-domain information extraction has been driven by the growth of Web-accessible data. We have staggering amounts of data from various structured and unstructured sources such as general Web text, online encyclopedias, query logs, web tables, or link anchor texts. Any proposed algorithm to extract information needs to harness several data sources and do it in a robust and scalable manner. Our work in this paper represents a first step towards that goal. In doing so, we achieved the following:

1. Improved coverage relative to a high accuracy instance-class extraction system while maintaining adequate precision.

2. Combined information from two different sources: free text and web tables.

3. Demonstrated a graph-based label propagation algorithm that given as little as five seeds per class achieved good results on a graph with more than a million nodes and 70 million edges.

In this paper, we started off with a simple graph. For future work, we plan to proceed along the following lines:

1. Encode richer relationships between nodes, for example instance-instance associations and other types of nodes.

2. Combine information from more data sources to answer the question of whether more data or diverse sources are more effective in increasing precision and coverage.

3. Apply similar ideas to other information extraction tasks such as relation extraction.

Acknowledgments

We would like to thank D. Sivakumar for useful discussions and the anonymous reviewers for helpful comments.

References

A. Azran. 2007. The rendezvous algorithm: multiclass semi-supervised learning with markov random walks. *Proceedings of the 24th international conference on Machine learning*, pages 49–56.

S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly. 2008. Video suggestion and discovery for youtube: taking random walks through the view graph.

K. Bellare, P. Talukdar, G. Kumaran, F. Pereira, M. Liberman, A. McCallum, and M. Dredze. 2007. Lightly-Supervised Attribute Extraction. *NIPS 2007 Workshop on Machine Learning for Web Search*.

M. Cafarella, A. Halevy, D. Wang, E. Wu, and Y. Zhang. 2008. Webtables: Exploring the power of tables on the web. *VLDB*.

C. Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database and Some of its Applications*. MIT Press.

M. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-92)*, pages 539–545, Nantes, France.

P. Indyk and J. Matousek. 2004. Low-distortion embeddings of finite metric spaces. *Handbook of Discrete and Computational Geometry*.

B. Jansen, A. Spink, and T. Saracevic. 2000. Real life, real users, and real needs: a study and analysis of user queries on the Web. *Information Processing and Management*, 36(2):207–227.

D. Lin and P. Pantel. 2002. Concept discovery from text. In *Proceedings of the 19th International Conference on Computational linguistics (COLING-02)*, pages 1–7.

K. McCarthy and W. Lehnert. 1995. Using decision trees for coreference resolution. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1050–1055, Montreal, Quebec.

E. Riloff and R. Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI-99)*, pages 474–479, Orlando, Florida.

M. Stevenson and R. Gaizauskas. 2000. Using corpus-derived name lists for named entity recognition. In *Proceedings of the 6th Conference on Applied Natural Language Processing (ANLP-00)*, Seattle, Washington.

M. Szummer and T. Jaakkola. 2002. Partially labeled classification with markov random walks. *Advances in Neural Information Processing Systems 14: Proceedings of the 2002 NIPS Conference*.

B. Van Durme and M. Paşca. 2008. Finding cars, goddesses and enzymes: Parametrizable acquisition of labeled instances for open-domain information extraction. *Twenty-Third AAAI Conference on Artificial Intelligence*.

R. Wang and W. Cohen. 2007. Language-Independent Set Expansion of Named Entities Using the Web. *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 342–350.

X. Zhu, Z. Ghahramani, and J. Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. *ICML-03, 20th International Conference on Machine Learning*.

Seeded Discovery of Base Relations in Large Corpora

Nicholas Andrews
BBN Technologies*
noa@bbn.com

Naren Ramakrishnan
Virginia Tech
naren@cs.vt.edu

Abstract

Relationship discovery is the task of identifying salient relationships between named entities in text. We propose novel approaches for two sub-tasks of the problem: identifying the entities of interest, and partitioning and describing the relations based on their semantics. In particular, we show that term frequency patterns can be used effectively instead of supervised NER, and that the p-median clustering objective function naturally uncovers relation exemplars appropriate for describing the partitioning. Furthermore, we introduce a novel application of relationship discovery: the unsupervised identification of protein-protein interaction phrases.

1 Introduction

Relationship extraction (RE) is the task of extracting named relationships between entities in text given some information about the relationships of interest. Relationship discovery (RD), on the other hand, is the task of finding which relations exist in a corpus without any prior knowledge. The discovered relationships can then be used to bootstrap RE, which is why RD has also been called unsupervised relation extraction (Rosenfeld and Feldman, 2006). RD generally involves three sub-tasks: entities of interest are either supplied or recognized in the corpus; second, of all phrases in which entities co-occur, those which express a relation are picked out; finally, these relationship phrases are partitioned based on their semantics and described. This work considers only binary relations (those between exactly two entities).

Finding entities of interest has involved either named entity recognition (NER) or general noun phrase (NP) chunking, to create the initial pool of candidate entities. In Section 2, we describe a corpus statistics approach, previously applied for web mining (Davidov and Rappoport, 2006), which we extend for relation discovery. Unlike supervised machine learning methods, this algorithm does not need training, is computationally efficient, and requires as input only the raw corpus and a small set of seed entities (as few as two). The result is a set of entities likely to be related to the seeds.

An assumption commonly held in RD work is that frequently co-occurring entity tuples are likely to stand in some fixed relation (Hasegawa et al., 2004; Shinyama and Sekine, 2006; Rosenfeld and Feldman, 2006; Rosenfeld and Feldman, 2007). Tuples which share similar contexts (the exact definition of context varies) are then grouped together in clusters of relations using variants of hierarchical agglomerate clustering (HAC). However, to our knowledge, no prior work has satisfactorily addressed the problem of describing the resulting clusters. In Section 3, we propose an approach which incorporates this requirement directly into the clustering objective: to find relation clusters which are well-described by a single exemplar.

In Section 4, we apply RD to recognize protein-protein interaction (PPI) sentences, using proteins as seeds for the entity discovery phase. We compare our results against special-purpose methods in terms of precision and recall on standard data sets.

The remainder of this paper is outlined below: Section 2 describes how a small number of input words (the entities of interest) are used as seeds for unsupervised entity discovery. Section 3 describes how discovered entities are used to discover relationships. Section 4 describes evaluation methodology and results. Section 5 describes related work. Section 6 concludes and discusses

*This work was conducted while author was at Virginia Tech.

directions for future work.

2 Entity discovery

For a corpus C, each sentence $s \in C$ with words $s = (w_1, w_2, ..., w_n)$, is mapped to the sequence $s' = f(s)$. The function f maps each word $w \in s$ to a symbol based on its frequency in C as follows:

$$f(w) = \begin{cases} S & \text{if } w \text{ is a seed word} \\ H & \text{otherwise if } w \text{ is a frequent word} \\ X & \text{otherwise} \end{cases}$$

For example, the sentence:

A and B are usually mediated by an overproduced C.

might be mapped to the sequence $(S, H, X, H, H, X, H, H, X, X)$, which we will write as $SHXHHXHHXX$ for brevity. In this case, A is a seed term, while B and C are not. The underlying assumption is that content words can be distinguished from other words based on their frequency in the corpus.

2.1 Pattern induction

In the example sentence, 'A and B are usually mediated by an overproduced C', 'and' is a good indicator that A,B share some aspect of their semantics; in this case, that they are both mediated by an overproduced C, and are therefore also likely to belong to same family or type of entities. The indicators 'and' and 'or' have together been used to discover word categories in lexical acquisition (Dorow et al., 2005). However, there can be many other such indicators, many discourse or corpus specific. To discover them, we use a slightly modified version of the method presented in (Davidov and Rappoport, 2006). In particular, in this work we consider named entities of arbitrary length (i.e., longer than a single token).

The corpus is searched for all instances of the frequency pattern $H_1\mathbf{S_1}H_2\mathbf{S_2}H_3$, for seed words S_1, S_2, and pattern (H_1, H_2, H_3). Of all these pattern instances, we keep those which also appear as $H_1\mathbf{S_2}H_2\mathbf{S_1}H_3$. If seed words appear on either side of the pattern, it is a good indication that the symmetric pattern expresses some sort of a conjunction, often domain specific. This procedure is repeated for variations of $HSHSH$ with the goal of capturing different forms of speech; for example, $HSHSH$ will capture '; A , B and', while $HSHHSH$ will capture '; A but not B ,' and so on. We enforce that frequent words appear before and after (i.e., surround) the two seed words to ensure they are standalone entities, and not part of a longer noun phrase. For example, the phrase 'IFN-gamma mRNA and IL-6 are' maps to $XXHSH$, and therefore 'mRNA' would (correctly) not be added to the entity pool.

New entities are added to the initial set of seed by matching symmetric patterns. If a seed word S is found to occur with an infrequent word X in any discovered symmetric pattern (as $H\mathbf{S}HXH$ or $HXH\mathbf{S}H$), then we add X to the pool of entities. This process can be bootstrapped as needed.

2.2 Chunking

In Section 3.1, sentences in which entities co-occur are clustered based on a measure of pairwise similarity. The features used in this similarity calculation are based on the surrounding or connecting words in the sentence in which entities co-occur. To ensure the context is not polluted with words which actually belong the entity NP (such as 'IFN-gamma *mRNA*') rather than the context, we use frequency patterns to search the corpus for common NP chunks.

In each sentence in which entities occur, we form a candidate chunk by matching the regular expression HX^*SX^*H, which returns all content-words X bracketing the entity S. Of all candidate chunks, we keep those which occur frequently enough to significantly affect the similarity calculations. The remaining chunks are pruned based on the entropy of the words appearing immediately before and after the chunk in the corpus; if a given chunk appears in a variety of contexts, it is more likely to express a meaningful collocation (Shimohata et al., 1997). Therefore, as an efficient filter on the candidate chunks, we discard those which tend to occur in the same contexts (where the context is $H...H$).

3 Identifying relation phrases

Once the pool of entities has been recognized in the corpus, those which frequently co-occur are taken as likely to stand in a relation. Order matters in that $S_1..S_2$ is considered a different entity co-occurrence (and therefore potential relation) than $S_2..S_1$. The effect of the co-occurrence threshold on the resulting relations is investigated in Section 4.

3.1 Clustering relation phrases

Partitioning the candidate relationships serves to identify groups of differently expressed relationships of similar semantics. The resulting clusters should cover the most important relations in a corpus between the entities of interest. The phrases in each cluster are expected to capture most syntactic variation in the expression of a given relationship. Therefore, the largest clusters are well suited as positive examples for training a relationship extractor (Rosenfeld and Feldman, 2006).

We take the context of a co-occurring tuple to be the terms connecting the two entities within the sentence in which they appear, and call the connecting terms a relation phrase (RP). Each RP is treated separately in the similarity calculations and the clustering. Relations are modeled using a vector space model. Each relation is treated as a vector of term frequencies (tf) weighted by tf $\times$ idf. RPs are preprocessed by filtering stopwords[1]. However, we do not stem the remaining words, as suffixes can be highly discriminative in determining the semantics of a relation (e.g., 'production' vs 'produced'). After normalizing vectors to unit length, we compute a similarity matrix by computing the dot product between the vectors for each distinct RP pair. The similarity matrix is then used as input for the clustering.

3.2 p-Median clustering

Prior approaches to relationship discovery have used HAC to identify relation clusters. HAC is attractive in unsupervised applications since the number of clusters is not required *a priori*, but can be determined from the resulting dendogram. On the other hand, a typical HAC implementation runs in $\Theta(N^2 \log(N))$, which can be prohibitive on larger data sets[2].

A further feature of HAC, and many other partitional clustering algorithms such as k-means and spectral cuts, is that the resulting clusters are not necessarily well-described by single instance. Relations, however, typically have a base or root form which would be desirable to uncover to describe the relation clusters. For example, in the following RPs:

induced transient increases in
induced biphasic increases in
induced an increase in
induced an increase in both
induced a further increase in

the phrase 'induced an increase in' is well suited as a base form of the relation and a descriptor for the cluster. The p-median clustering objective is to find p clusters which are well-described by a single exemplar. Formally, given an $N \times N$ similarity matrix, the goal is to select p columns such that the sum of the maximum values within each row of the selected columns are maximized.

Note that an exemplar can also be chosen *a posteriori* using some heuristic; for example, the most frequently occurring instance in a cluster can be taken as the exemplar. However, the p-median clustering objective is robust, and ensures that only those clusters which are well described by a single exemplar appear in the resulting partition of the relations. This means that the optimal number of clusters for the p-median clustering objective in a given data set will usually be quite different (usually higher) than the optimal number of groups according to the HAC, k-means, or normalized cut objectives.

Affinity propagation (AP) is the most efficient approximation for the p-median problem that we are aware of, which also has the property of not requiring the number of clusters as an explicit input (Frey and Dueck, 2007). Runtime is linear in the number of similarities, which in the worst case is N^2 (for N relations), but in practice many relations share no words in common, and therefore do not need to have their similarity considered in the clustering.

AP is an iterative message-passing procedure in which the objects being clustered compete to serve as cluster exemplars by exchanging two types of messages. The responsibility $r(x,m)$, sent from object $x \in \mathcal{X}$ (for set $\mathcal{X}$ of objects to be clustered) to candidate exemplar $m \in \mathcal{X}$, denotes how well-suited m is of being the exemplar for x by considering all other potential exemplars m' of x:

$$s(x,m) - \max_{m' \in \mathcal{X}, m' \neq m} a(x,m') + s(x,m')$$

where $s(x,m)$ is the similarity between x, m. The availability $a(x,m)$ of each object $x \in \mathcal{X}$ is initially set to zero. Availabilities, sent from candidate exemplar m to object x, increase as evidence for m to serve as the exemplar for x increases:

$$\min\left\{0, r(m,m) + \sum_{x' \in \mathcal{X}, x' \notin \{x,m\}} \max\{0, r(x',m)\}\right\}$$

Each object to be clustered is assigned an initial preference of becoming a cluster exemplar. If there are no *a priori* preferences for cluster exemplars, the preferences are set to the median similarity (which can be thought of as the 'knee' of the objective function graph vs. number of clusters), and exemplars emerge from the message passing procedure. However, shorter RP are more likely to contain base

[1]We use the English stopword list from the Snowball project, available at `http://snowball.tartarus.org/`

[2]An optimization to $\Theta(N^2)$ is possible for single-linkage HAC.

forms of relations (because longer phrases likely contain additional words specific to the sentence). Therefore, we include a slight scaling factor in the preferences, which assigns shorter RP higher initial values (up to $1.5\times$ the median similarity).

3.3 Pruning clusters

After clustering relation phrases with AP, we prune the resulting partition by evaluating the number of different relation instances appearing in each cluster, as well as the entities involved. In our experiments, we discard all clusters smaller than a certain threshold, since we ultimately wish to use the clustering to train RE, and small clusters do not provide enough positive examples for training (we investigate the effect of this threshold in Section 4.2). We further assume that for a relationship to be useful, a number of different entities should stand in this relation. In particular, we inspect the set of left and right arguments in the cluster, which (in English) usually correspond to the subject and object of the sentence. If a single entity constitutes more than two thirds ($\frac{2}{3}$) of the left or right arguments of a cluster, then this cluster is discarded from the results. Our assumption is that these clusters describe relations too specific to be useful.

4 Evaluation

RD systems are usually evaluated based on their results for a particular task such as RE (Rosenfeld and Feldman, 2006), or by a manual inspection of their results (Davidov et al., 2007; Rosenfeld and Feldman, 2007; Hasegawa et al., 2004), but we are not aware of any which examines the effects of parameters on performance exhaustively. In this section we test several hypotheses of RD using data sets which are already labeled for sentences which contain entities of a particular type and in a fixed relation of some kind. In particular, we adapt the output of the discovery phase to identify phrases which express PPIs. While this task is traditionally performed using supervised algorithms such as support vector machines (Erkan et al., 2007), we show that RD is capable of achieving similar levels of precision without any manually annotated training data.

4.1 Method

We construct a corpus of 87300 abstracts by querying the PubMed database with the proteins shown in Table 1. The 60 most frequent words are considered definite non-entities; all remaining words are candidate entities. This corpus serves as input for the relationship discovery. As seeds, we use the same 25 proteins used to query the database. Since all seeds are proteins, we expect the entities discovered to be proteins. The pattern induction found roughly 200 symmetric extraction patterns, which yield 4402 unique entities after 1 pass through the corpus. Depending on the frequency of the seeds in the corpus, more passes through the corpus might be needed (bootstrapping with the discovered entities after each pass). We retain all chunks that appear at least 10 times in the corpus, yielding 3282 additional entities after entropy pruning.

Table 1: Proteins queried to create the evaluation corpus.

Seed entities (proteins)				
c-cbl	AmpC	CD18	CD54	CD5
CD59	CK	c-myc	CNP	DM
EBNA	GSH	IL-8	IL-1beta	JNK1
p38	PABP	PCNA	PP1	PP2a
PPAR	PSM	TAT	TNF-alpha	TPO

A PPI denotes a broad class of bio-medical relationships between two proteins. One example of an interaction is where the two proteins bind together to form a structural complex of cellular machinery such as signal transduction machinery. A second example is when one protein binds upstream of the DNA sequence encoding a gene which encodes the second protein. A final example is when proteins serve as enzymes catalyzing successive steps of a biochemical reaction. More categories of interactions are continually being catalogued and hence unsupervised identification of PPIs is important in biomedical text mining.

4.2 Experiment 1: PPI sentence identification

Method: To evaluate the performance of our system, we measure how well the relationships discovered compare with manually selected PPI sentences. To do so, we follow the same procedure and data sets used to evaluate semi-supervised classification of PPI sentences (Erkan et al., 2007). The two data sets are AIMED and CB, which have been marked for protein entities and interaction phrases[3].

For each sentence in which n proteins appear, we build $\binom{n}{2}$ phrases. Each phrase consists of the words between each entity combination, and is labeled as positive if it describes a PPI, or negative otherwise. This results in 4026 phrases for the

[3]Available in preprocessed form at `http://belabog.si.umich.edu/biocreative`

AIMED data set (951 positive, 3075 negative), and 4056 phrases for the CB data set (2202 positive, 1854 negative).

The output of the discovery phase is a clustering of RPs. For purpose of this experiment, we ignore the partition and treat the phrases in aggregate. A phrase in the evaluation data set is classified as positive (describing a PPI) if any substring of the phrase matches an RP in our output. For example, if the phrase is:

A significantly inhibited B

and the string 'inhibited' appears as a relation in our output, then this phrase is marked positive. Otherwise, the phrase is marked negative.

Performance is evaluated using standard metrics of precision (P), recall (R), and F-measure (F_1), defined as:

$$P = \frac{TP}{TP + FP}; \quad R = \frac{TP}{TP + FN}$$

where TP is the number of phrases correctly identified as describing a PPI, FP is the number of phrases incorrectly classified as describing a relation, and FN is the number of interaction phrases (positives) marked negative. F_1 is defined as:

$$F_1 = \frac{2PR}{P + R}$$

We calculate P, R, and F_1 for three parameters affecting which phrases are identified as expressing a relation:

- the minimum co-occurrence threshold that controls which entity tuples are kept as likely to stand in some fixed relation
- the minimum cluster size that controls which groups of relations are discarded
- the minimum RP length that controls the smallest number of words appearing in relations

The threshold on the length of the relations can be thought of as controlling the amount of contextual information expressed. A single term relation will be very general, while longer RPs express a relation very specific to the context in which they are written. The results are reported in Figures 1 through 6. Odd numbered figures use the AIMED corpus; even numbered figures the CB corpus. **Results:** Discarding clusters below a certain size had no significant effect on precision. However, this step is still necessary for bootstrapping RE, since machine learning approaches require a sufficient number of positive examples to train the extractor.

Table 2: Comparison with supervised methods–AIMED corpus

Method	P	R	F_1
RD-F_1	30.08	60.67	40.22
RD-P	**55.17**	5.04	9.25
(Yakushiji et al., 2005)	33.70	33.10	33.40
(Mitsumori et al., 2006)	54.20	42.60	47.70
(Erkan et al., 2007)	**59.59**	60.68	59.96

Table 3: Comparison with supervised methods–CB corpus

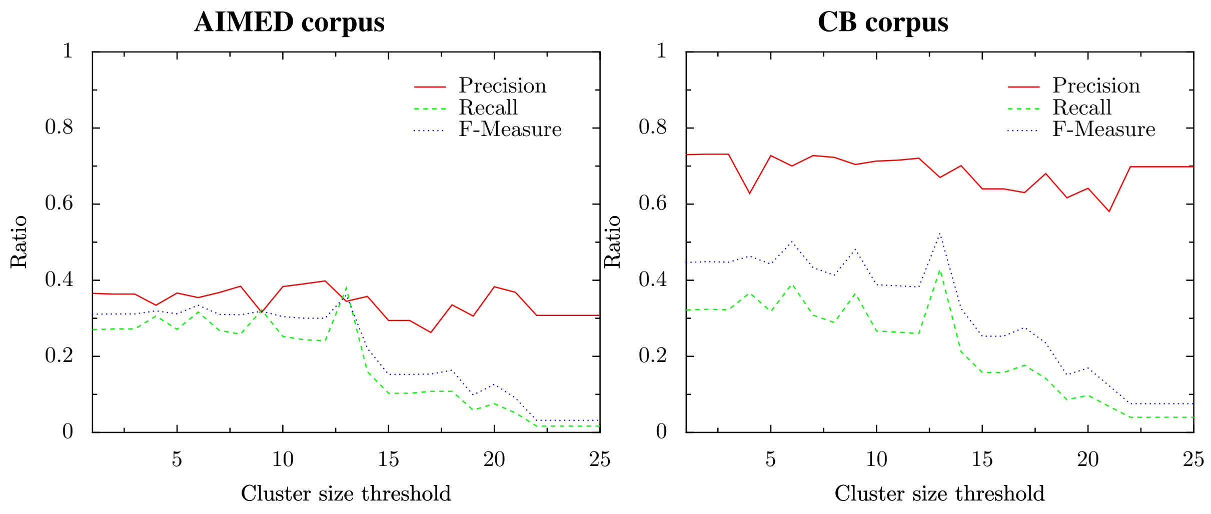
Method	P	R	F_1
RD-F_1	65.03	69.16	67.03
RD-P	**86.27**	2.00	3.91
(Erkan et al., 2007)	**85.62**	84.89	85.22

On the other hand, our results confirm the observation that frequently co-occurring pairs of entities are likely to stand in a fixed relation. On the CB corpus, precision ranges from 0.63 to 0.86 for phrases between entities co-occurring at least 50 times. On the AIMED corpus, precision ranges from 0.29 to 0.55 in the same threshold range.

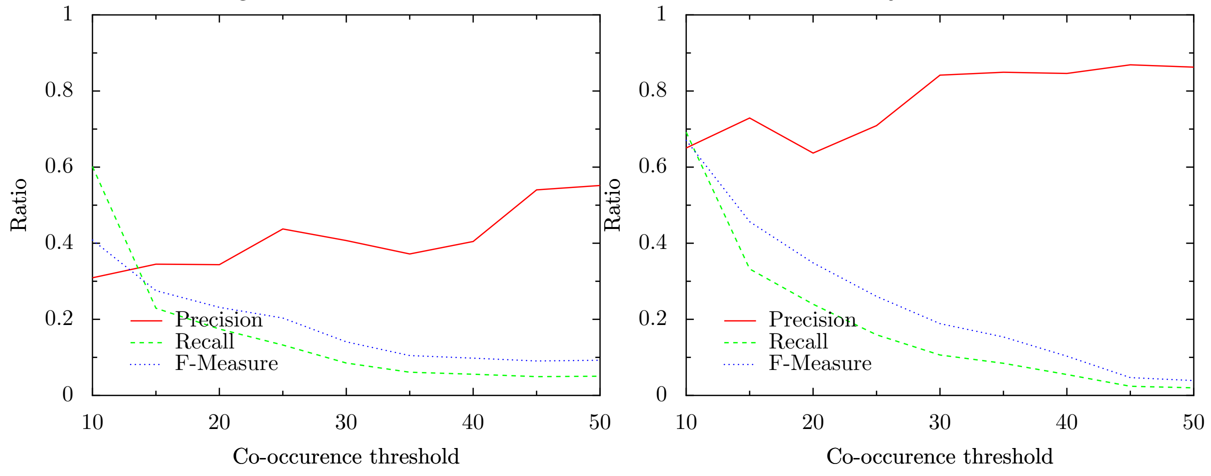
The minimum phrase length had the most impact on performance, which was particularly evident in the CB corpus: this corpus reached perfect precision discarding all RPs of fewer than 3 words. Lower thresholds result in significantly more relations, at the cost of precision.

The generally lower performance on the AIMED corpus suggests that our training data (retrieved from the seed proteins) provided less coverage for those interactions than for the those in the CB corpus.

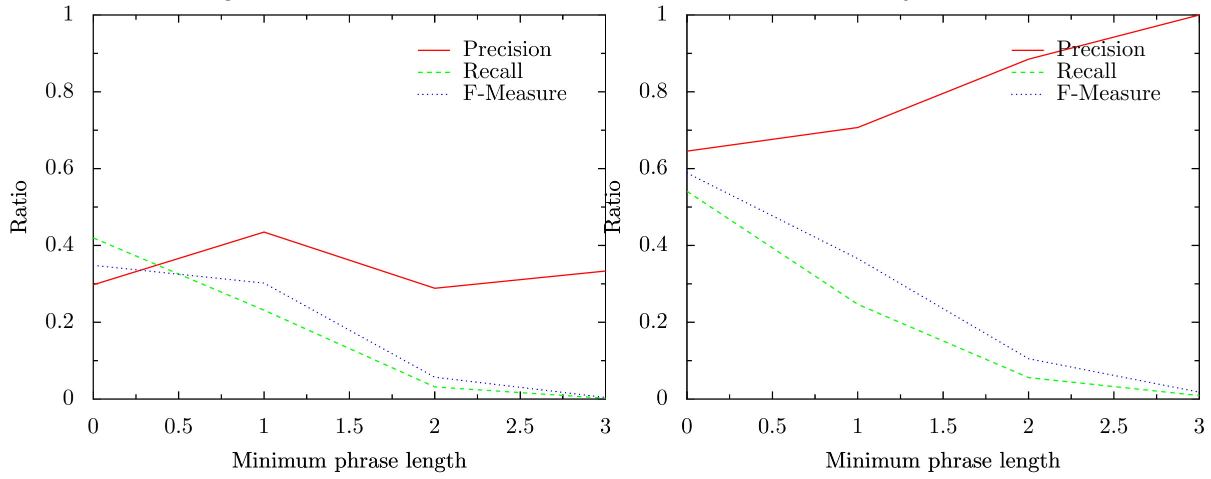
Table 2 and Table 3 compare our results at fixed parameter settings with supervised approaches. RD-F_1 reports parameters which give highest recall and RD-P highest precision. Specifically, both RD-F_1 and RD-P use a minimum RP length of 1, RD-F_1 uses a co-occurrence threshold of 10, and RD-P uses a co-occurrence threshold of 50. As expected, RD alone does not match combined precision and recall of state-of-the-art supervised systems. However, we show better performance than expected. RD-F_1 outperforms the best results of (Yakushiji et al., 2005). RD-P settings outperform or match the precision of top-performing systems on both datasets.



Figures 1 & 2: Performance as minimum cluster size is adjusted



Figures 3 & 4: Performance as co-occurrence threshold is adjusted



Figures 5 & 6: Performance as minimum phrase length is adjusted

4.3 Experiment 2: clustering relations

Method: We evaluate the appropriateness of the p-median clustering as follows. For each cluster, we take the cluster exemplar as defining the base relation. If the base relation does not express something meaningful, then we mark each member of the cluster incorrect. Otherwise, we label each member of the cluster either as semantically similar to the exemplar (correct) or different than the exemplar (incorrect). Thus, clusters with inappropriate exemplars are heavily penalized. These results are reported in Table 4. For purpose of this experiment, we use the same parameters as for RD-P, and evaluate the 20 largest clusters.

Results: In the 20 largest clusters, each cluster exemplar expressed something meaningful. 3 of the cluster exemplars were not representative of their other members. We found that most error was due to stopwords not being considered in our similarity calculations. For example, 'detected by' and 'detected in' express the same relationship in our similarity calculations; however, they are clearly quite different. Another source of error evident in Table 4 are mistakes in the pattern and entropy based chunking. The exemplar 'mrna expression in' includes the token 'mrna', which belongs with the left protein NP in the relation chosen as an exemplar.

Table 4: Base relations identified using RP-P parameters

Exemplar	**Size**	P **(%)**
by activation of	33	87.9
was associated with	28	92.9
was induced by	24	83.3
was detected by	24	83.3
as compared with the	25	92.0
were measured with	23	87.0
mrna expression in	21	**9.5**
in response to	21	95.23
was determined by	21	90.4
with its effect in	19	**10.5**
was correlated with	18	100.0
by induction of	16	93.8
for binding to	16	75.0
is mediated by	16	93.8
was observed by	16	50.0
is an important	15	66.6
increased expression of	15	60.0
related to the	15	93.3
protein production as well as	15	**33.3**
dependent on	14	85.7
Median precision: 86.35		

5 Related work

RD is a relatively new area of research. Existing methods differ primarily in the amount of supervision required and in how contextual features are defined and used.

(Hasegawa et al., 2004) use NER to identify frequently co-occurring entities as likely relation phrases. As in this work, they use the vector model and cosine similarity to define a measure of similarity between relations, but build relation vectors out of *all* instances of each frequently co-occurring entity pair. Therefore, each mention of the same co-occurring pair is assumed to express the same relationship. These aggregate feature vectors are clustered using complete-linkage HAC, and cluster exemplars are determined by manual inspection for evaluation purposes. (Shinyama and Sekine, 2006) rely further on supervised methods, defining features over a full syntactic parse, and exploit multiple descriptions of the same event in newswire to identify useful relations.

(Rosenfeld and Feldman, 2006) consider the use of RD for unsupervised relation extraction, and use a more complex pattern-learning approach to define feature vectors to cluster candidate relations, reporting gains in accuracy compared with the tf $\times$ idf weighed features used in (Hasegawa et al., 2004) and in this work. They also use HAC, and do not address the description of the relations. Arbitrary noun phrases obtained through shallow parsing are used as entities. (Rosenfeld and Feldman, 2007) use a feature ranking scheme using separability-based scores, and compare the performance of different variants of HAC (finding single-linkage to perform best). The complexity of the feature ranking-scheme described can be greater than the clustering itself; in contrast, while we use simple features, our approach is much more efficient.

(Davidov et al., 2007) introduce the use of term frequency patterns for relationship discovery. However, they search for a specific type of relationship; namely, attributes common to all entities of a particular type (for example, all countries have the attribute *capital*), and use a special purpose set of filters rather than entity co-occurrence and clustering. Our work can be seen as a generalization of theirs to relationships of any kind, and we extend the use of frequency patterns to finding general n-gram entities rather than single word entities.

(Madkour et al., 2007) give an excellent overview of biomedical NER and RE. They propose a statistical system for RE, but rely on NER, POS tagging, and the creation of a dictionary for each domain of application. Also, they do not cluster relationships into semantically related groups.

6 Conclusion

Our work makes a series of important improvements to the state-of-the-art in relationship discovery. First, by incorporating entity discovery into the relationship discovery pipeline, our method does not require distinct training phases to accommodate different entity types, relations, or discourse types. Second, p-median clustering effectively uncovers the base form of relations present in the corpus, addressing an important limitation in usability. In terms of specific hypotheses, we have tested and confirmed that co-occurrence can be a good indicator of the presence of a relationship but the size of a cluster is not necessarily a good indicator of the importance or strength of the discovered relationship. Furthermore, we have shown that longer RPs with more context give higher precision (at the cost of reduced coverage). Finally, the integration of ideas in our approach—unsupervisedness, efficiency, flexibility (in application), and specificity—is novel in itself.

In future work, we seek to expand upon our RD methods in three directions. First, we would like to generalize the scope of our discovery pipeline beyond binary relations and with richer considerations of context, even across sentences. Second, we hope to achieve greater tunability of performance, to account for additional discovery metrics besides precision. Finally, we intend to induce entire concept maps from text using the discovered relations to bootstrap an RE phase, where the underlying problem is not just of inferring multiple types of relations, but to have sufficient co-ordination among the discovered relations to ensure connectedness among the resulting concepts.

While our method requires no supervision in the form of manually annotated entities or relations, the effectiveness of the system relies on the careful tuning of a number of parameters. Nevertheless, the results reported in Section 4.2 suggest that the two parameters that most significantly affect performance exhibit predictable precision/recall behavior. Of the parameters not considered in Section 4.2, we would like to further investigate the benefits of chunking entities on the resulting base relations, experimenting with different measures of collocation.

Acknowledgements

We would like to thank our anonymous reviewers for their thought-provoking questions. This work was supported in part by the Institute for Critical Technology and Applied Science (ICTAS), Virginia Tech.

References

Dmitry Davidov and Ari Rappoport. 2006. Efficient unsupervised discovery of word categories using symmetric patterns and high frequency words. In *ACL '06: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 297–304, Morristown, NJ, USA. Association for Computational Linguistics.

Dmitry Davidov, Ari Rappoport, and Moshe Koppel. 2007. Fully unsupervised discovery of concept-specific relationships by web mining. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 232–239, Prague, Czech Republic, June. Association for Computational Linguistics.

Beate Dorow, Dominic Widdows, Katarina Ling, Jean-Pierre Eckmann, Danilo Sergi, and Elisha Moses. 2005. Using curvature and markov clustering in graphs for lexical acquisition and word sense discrimination. In *MEANING 05: 2nd workshop organized by the MEANING Project*, Trento, Italy, February.

Gunes Erkan, Arzucan Ozgur, and Dragomir R. Radev. 2007. Semi-supervised classification for extracting protein interaction sentences using dependency parsing. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 228–237.

Brendan J. Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *Science*, 315:972–976.

Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. 2004. Discovering relations among named entities from large corpora. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 415, Morristown, NJ, USA. Association for Computational Linguistics.

Amgad Madkour, Kareem Darwish, Hany Hassan, Ahmed Hassan, and Ossama Emam. 2007. Bionoculars: Extracting protein-protein interactions from biomedical text. In *Biological, translational, and clinical language processing*, pages 89–96, Prague, Czech Republic, June. Association for Computational Linguistics.

T. Mitsumori, M. Murata, Y. Fukuda, K. Doi, and H. Doi. 2006. Extracting protein-protein interaction information from biomedical text with svm. *IEICE Transactions on Information and Systems*, 89(8):2464–2466.

Benjamin Rosenfeld and Ronen Feldman. 2006. High-performance unsupervised relation extraction from large corpora. In *ICDM '06: Proceedings of the Sixth International Conference on Data Mining*, pages 1032–1037, Washington, DC, USA. IEEE Computer Society.

Benjamin Rosenfeld and Ronen Feldman. 2007. Clustering for unsupervised relation identification. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 411–418, New York, NY, USA. ACM.

Sayori Shimohata, Toshiyuki Sugio, and Junji Nagata. 1997. Retrieving collocations by co-occurrences and word order constraints. In *In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 476–481.

Yusuke Shinyama and Satoshi Sekine. 2006. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 304–311, New York City, USA, June. Association for Computational Linguistics.

A. Yakushiji, Y. Miyao, Y. Tateisi, and J. Tsujii. 2005. Biomedical information extraction with predicate-argument structure patterns. In *Proceedings of the eleventh annual meeting of the association for natural language processing*, pages 93–96.

Mention Detection Crossing the Language Barrier

Imed Zitouni and Radu Florian
IBM T.J. Watson Research Center
1101 Kitchawan Rd, Yorktown Heights, NY 10598
{izitouni, raduf}@us.ibm.com

Abstract

While significant effort has been put into annotating linguistic resources for several languages, there are still many left that have only small amounts of such resources. This paper investigates a method of propagating information (specifically mention detection information) into such low resource languages from richer ones. Experiments run on three language pairs (Arabic-English, Chinese-English, and Spanish-English) show that one can achieve relatively decent performance by propagating information from a language with richer resources such as English into a foreign language alone (no resources or models in the foreign language). Furthermore, while examining the performance using various degrees of linguistic information in a statistical framework, results show that propagated features from English help improve the source-language system performance even when used in conjunction with all feature types built from the source language. The experiments also show that using propagated features in conjunction with lexically-derived features only (as can be obtained directly from a mention annotated corpus) yields similar performance to using feature types derived from many linguistic resources.

1 Introduction

Information extraction is a crucial step toward understanding a text, as it identifies the important conceptual objects and relations between them in a discourse. It includes classification, filtering, and selection based on the language content of the source data, i.e., based on the meaning conveyed by the data. It is a crucial step for several applications, such as summarization, information retrieval, data mining, question answering, language understanding, etc. This paper addresses an important and basic task of information extraction: *mention detection*[1]: the identification and classification of textual references to objects/abstractions *mentions*, which can be either named (e.g. John Smith), nominal (the president) or pronominal (e.g. he, she). For instance, in the sentence

> <u>President</u> <u>*John Smith*</u> said <u>*he*</u> has no comments.

there are three mentions: *President*, *John Smith* and *he*. This is similar to the named entity recognition (NER) task with the additional twist of also identifying nominal and pronominal mentions.

A few languages have received a lot of attention in terms of natural language resources that were created – for instance, in English one has access to labeled part-of-speech data, word sense information, parse tree structure, discourse, semantic role labeles, named entity data, to name just a few (our apologies if we missed your favorite resource). There are a few other languages that also have annotated resources (such as Arabic, Chinese, German, French, Spanish, etc), but also a very large number of languages with few resources. It would be very useful if one could make use of the resources in the former languages to help bootstrapping (or just the projection) of resource in any resource-challenged language.

Information transfer from a language to another can be very useful when the "donor" language has more resources than the receiving one. As resources grow in quantity and quality in the receiving language, it becomes less and less likely that there will be a gain in performance by transfering information, as there are several sources of noise involved in the

[1]We adopt here the ACE (NIST, 2007) nomenclature

process - such as the translation (machine generated or not) and the inherent imperfection of the mention detection in the donor language. To test this hypothesis, we conducted experiments on systems build with a varied amount of resources in the receiving language, starting with the case where there are none[2] (all information is transferred through translation alignment), and ending with the case where we used all the resources we could gather for that language. The experiments will show that the gain in performance decreases with the amount of resources used in the source language, but, still, even when all resources were used, a statistically significant gain was still observed.

Similarly to classical NLP tasks such as text chunking (Ramshaw and Marcus, 1995) and named entity recognition (Tjong Kim Sang, 2002), we formulate mention detection as a sequence classification problem, by assigning a label to each token in the text, indicating whether it starts a specific mention, is inside a specific mention, or is outside any mentions. The classification is performed with a statistical approach, built around the maximum entropy (MaxEnt) principle (Berger et al., 1996), that has the advantage of combining arbitrary types of information in making a classification decision.

2 Previous Work

There are several investigations in literature that explore using parallel corpora to transfer information content from one language (most of the time English) to another. The earliest investigations of the subject have been performed, on word sense disambiguation (Dagan et al., 1991; P.F.Brown et al., 1991; Gale et al., 1992) (perhaps unsurprisingly given its close connection to machine translation) – all propose and (lightly) evaluate methods to use word sense information extracted from the target language to help the sense resolution in the source language and machine translation. (Dagan and Itai, 1994) explicitly suggests performing word sense disambiguation in the target language (English in the article) with the goal of resolving ambiguity in the source language (Hebrew), and show moderate improvement on a small data set[3]. More recently, (Diab and Resnik, 2001) presents a method for performing word sense tagging in both the source and target texts of parallel bilingual corpora with the English WordNet sense inventory, by using translation correspondences.

On more general cross-language information transfer, (Yarowsky et al., 2001) proposed and evaluated a method of propagating POS tagging, named mention, base noun phrase, and morphological information from English into a foreign language, which is very similar to the one presented in this article (experiments were run on French, Chinese, Czech, and Spanish – on human-generated translations). Their results show a significant improvement in performance while building an automatic classifier on the projected annotations over the same automatic classifier trained on a small amount of annotated data in the source language. (Riloff et al., 2002) extends the ideas in (Yarowsky et al., 2001), by showing how it can be used, in conjunction with an automatically trained information extraction system on the source language, to bootstrap the annotation of resources in the target language. They show that they can obtain 48 F-measure on a information extraction task identifying locations, vehicles and victims in plane crashes. (Hwa et al., 2002) proposes a framework that enables the acquisition of syntactic dependency trees for low-resource languages by importing linguistic annotation from rich-resource languages (English). The authors run a large-scale experiment in which Chinese dependency parses were induced from English, and show that a parser trained on the resulting trees outperformed simple baselines. (Cabezas et al., 2001) investigates a similar method of propagating syntactic treebank-like annotations from English to Spanish.

Finally, a large body of research has been done on *cross-language information retrieval*, where the goal is to find information in one language (e.g. Chinese newswire) corresponding to a query in a different language (e.g. English) – although the list of relevant papers is too long to be mentioned here (see, for instance, (Grefenstette, 1998)).

The work presented here differs from the information extraction investigations presented above in two aspects:

- it handles unrestricted text and a full set of

[2]While applying this method in the case where the source language has absolutely no resources might be an interesting test case, we don't see it as being realistic. Resources are build nowadays in a large variety of languages, and not making use of them is rather foolish (a certain big bird and sand comes to mind).

[3]Very small by "modern" standards - 137 examples. Probably because at the time the article was written, there were no large publicly annotated databases, such as Semcor.

mention types (the ACE entity types) during the information transfer

- it investigates whether using a resource-rich language (English) can improve on the performance obtained by using various degrees of existent resources in the source language (Arabic, Chinese, Spanish)
- the information transfer is performed over machine generated translations and alignments.

3 Mention Detection

As mentioned in the introduction, the mention detection problem is formulated as a classification problem, by assigning to each token in the text a label, indicating whether it starts a specific mention, is inside a specific mention, or is outside any mentions.

Good performance in many natural language processing tasks has been shown to depend heavily on integrating many sources of information (Florian et al., 2004).[4] Given this observation, we are interested in algorithms that can easily integrate and make effective use of diverse input types. We select a exponential classifier, the Maximum Entropy (MaxEnt henceforth) classifier that integrates arbitrary types of information and makes a classification decision by aggregating all information available for a given classification. But the reader can replace it with her favorite feature-based classifier throughout the paper.

To help with the presentation, we introduce some notations: let $\mathcal{Y} = \{y_1, \ldots, y_n\}$ be the set of predicted classes, $\mathcal{X}$ be the example space and $\mathcal{F} = \{0,1\}^m$ be a feature space. Each example $x \in \mathcal{X}$ has associated a vector of m binary features $f(x) = (f_1(x), \ldots, f_m(x))$. The goal of the training process is to associate examples $x \in \mathcal{X}$ with either a probability distribution over the labels from $\mathcal{Y}$, $P(\cdot|x)$(if we are interested in *soft* classification) or associate one label $y \in \mathcal{Y}$ (if we are interested in *hard* classification).

The MaxEnt algorithm associates a set of weights $\{\alpha_{ij}\}_{j=1\ldots m}^{i=1\ldots n}$ with the features $(f_j)_i$, and computes the probability distribution as

$$P(y_i|x) = \frac{1}{Z(x)} \prod_{j=1}^{m} \alpha_{ij}^{f_j(x,y_i)}, \qquad (1)$$

$$Z(x) = \sum_i \prod_j \alpha_{ij}^{f_j(x,y_i)}$$

where $Z(x)$ is a normalization factor. The $\{\alpha_{ij}\}_{j=1\ldots m}$ weights are estimated during the training phase to maximize the likelihood of the data (Berger et al., 1996). In this paper, the MaxEnt model is trained using the *sequential conditional generalized iterative scaling* (SCGIS) technique (Goodman, 2002), and it uses a *Gaussian prior* for regularization (Chen and Rosenfeld, 2000).

Now take $x_1^N = (x_1, x_2, \ldots x_N)$, a sequence of contiguous tokens (i.e., a sentence or a document) in the source language. The goal of mention detection system is to find the most likely sequence of labels $y_1^N = (y_1, y_2 \ldots y_N)$ that best matches the input x_1^N. In the mention detection case, each token x_i in x_1^N is tagged with a label y_i as follows:[5]

- if it's not part of any entity, $y_i = O$ (O for "outside any mentions")
- if it is part of an entity, it is composed of a subtag specifying whether it starts a mention (*B*-) or is inside a mention (*I*-), and a sub-type corresponding to mention type (e.g. *B-PERSON*). In ACE, there are seven possible types: person, organization, location, facility, geopolitical entity (GPE), weapon, and vehicle.

To compute the best sequence y_1^N, we use

$$\begin{aligned} y_1^N &= \arg\max_{\hat{y}_1^N} P\left(\hat{y}_1^N | x_1^N\right) \\ &= \arg\max_{\hat{y}} \prod P\left(\hat{y}_j | x_1^N, \hat{y}_1^{j-1}\right) \\ &= \arg\max_{\hat{y}} \prod_j P\left(\hat{y}_j | x_1^N, y_{j-k}^{j-1}\right) \end{aligned}$$

where $P\left(\hat{y}_j | x_1^N, y_{j-k}^{j-1}\right)$ has an exponential form of the type (2). We also used the standard Markov assumption that the probability $P\left(\hat{y}_j | x_1^N, \hat{y}_1^{j-1}\right)$ only depends on the previous k classifications. This model is similar to the MEMM model (McCallum et al., 2000), but it does not separate the probability into generation probabilities and transition probabilities, and, crucially, has access to "future" observed features (i.e. it can examine the entire x_1^N sequence, though in practice it will only examine some small part of it) – which is one way of eliminating label

[4]In fact, the feature set used for classification has a much larger impact on the performance of the resulting system than the classifier method itself.

[5]The mention encoding is the IOB2 encoding presented in (Tjong Kim Sang and Veenstra, 1999) and introduced by (Ramshaw and Marcus, 1994) for base noun phrase chunking.

bias observed by (Lafferty et al., 2001).[6]

The experiments are run on four languages, part of the ACE-2007 evaluation (NIST, 2007): Arabic, Chinese, English and Spanish.[7] Systems across the languages use a large range of features, including lexical (words and morphs in a 3-word window, prefixes and suffixes of length up to 4 characters, WordNet (Miller, 1995) for English), syntactic (POS tags, text chunks), and the output of other information extraction models. These features were described in (Florian et al., 2004), and are not discussed here. In this paper we focus on the examining the benefit of cross-language mention propagation information in improving mention detection systems.

Besides generic types of features, we also have implemented language-specific features:

- In Arabic, blank-delimited words are composed of zero or more prefixes, followed by a stem and zero or more suffixes. Each prefix, stem or suffix is a token; any contiguous sequence of tokens can represent a mention. Similar to the approaches described in (Florian et al., 2004) and (Zitouni et al., 2005), we decided to "condition" the output of the system on the segmented data: the text is segmented first into tokens and classification is then performed on tokens. The segmentation model is similar to the one presented by (Lee et al., 2003) and obtains an accuracy of 98%.

- In Chinese text, unlike in Indo-European languages, words neither are white-space delimited nor do they have capitalization markers. Instead of a word-based model, we build a character-based one, since word segmentation errors can lead to irrecoverable mention detection errors; Jing et al. (2003) also observes that character-based models are better performing than word-based ones. Word segmentation information is still useful and is integrated as an additional feature stream.

- In English and in Spanish mention detection systems are similar to those described in (Florian et al., 2004) where words are the tokens to classify.

[6]In fact their example of label bias can be trivially solved by allowing the classifier to examine features for subsequent words.

[7]The ACE data has the nice property of being consistent in annotations across these languages.

4 Cross-Language Mention Propagation

The approach proposed in this article requires a mention detection system build in a resource-rich language, and a *translation* from the source language to the resource-rich language, together with *word alignment*. This assumption is realistic: while truly parallel data (humanly created) might be in short supply or harder to acquire, adapting statistical machine translation (SMT) systems from one language-pair to another is not as challenging as it used to be (Al-Onaizan and Papineni, 2006). We also find that there is a large number of parallel corpora available these days which cover many language pairs. For example, for the European Union's 23 official languages we find 253 language pairs; each document in one language might have to be translated in all other 22 languages. This is in addition to parallel corpora one could get from books, including religious texts such as the Bible, that are translated to a large number of languages. On the other hand, even though mention detection system is important for many natural language processing applications, we still find lack of mention-annotated corpora in many languages. In the approach we propose below, the annotated corpus used to train the mention detection classifier does not have to be part of a parallel corpus.

To start the process, we first use a SMT system to translate the source unit (document or sentence) x_1^N into the resource-rich language, yielding the sequence $\xi_1^M = (\xi_1, \xi_2, \ldots \xi_M)$. Taking the sequence of tokens ξ_1^M as input, the MaxEnt classifier assigns a mention label to each token, building the label sequence $\psi_1^M = (\psi_1, \psi_2 \ldots \psi_M)$. Using the SMT-produced word alignment between source text x_1^N and translated text ξ_1^M (Koehn, 2004),we propagate the target labels ψ_1^M to the source language building the label sequence $\tilde{y}_1^N = (\tilde{y}_1, \tilde{y}_2 \ldots \tilde{y}_N)$.[8] As an example, if a sequence of tokens in the resource-rich language $\xi_i \xi_{i+1} \xi_{i+2}$ is aligned to $x_j x_{j+1}$ in the source language and if $\xi_i \xi_{i+1} \xi_{i+2}$ is tagged as a location mention, then the sequence $x_j x_{j+1}$ can be labeled as a location mention: B-LOC, I-LOC. Hence, each token x_i in x_1^N is tagged with a corresponding propagated label $\tilde{y}_i$ in $\tilde{y}_1^N$, $\tilde{y}_i = \phi\left(i, A, \psi_1^M\right)$, where A is the alignment between the source and resource-rich languages. In cases when the alignment is 1-to-1 the function becomes the identity, but one can imagine different scenarios which can be used in

[8]Or by using Giza++ if your favorite engine does not give you word alignment.

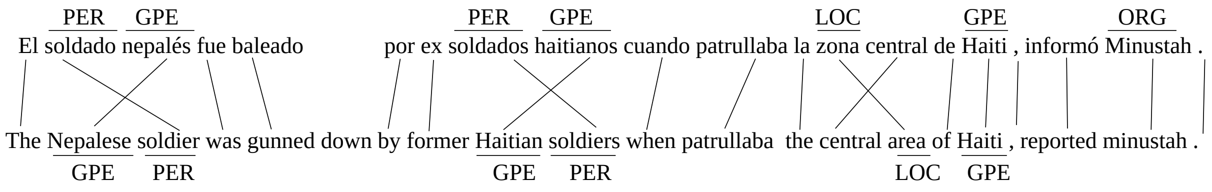

PER
GPE
El soldado nepalés fue baleado por ex soldados haitianos cuando patrullaba la zona central de Haiti , informó Minustah .
PER
GPE
LOC
GPE
ORG
The Nepalese soldier was gunned down by former Haitian soldiers when patrullaba the central area of Haiti , reported minustah .
GPE
PER
GPE
PER
LOC
GPE

Figure 1: Word alignment for a Spanish sentence and its English machine-translation. The mention labels shown are the gold-standard ones for Spanish and the automatically detected ones for English. If mentions were to be propagated from English to Spanish, the last mention would be a miss, due to the fact that the English mention detection failed to identify 'minustah' as an organization.

many-to-many alignment cases. The alignement we use in this paper is 1-to-many ($\{1...n\}$) from the source language (eg., Arabic) to the resource-rich language (e.g., English). Once we use SMT word alignment to propagate label sequence ψ_1^M of ξ_1^M to the corresponding text x_1^N in the target language, we end up with a sequence of labels $\tilde{y}_1^N$ where for each token x_i in x_1^N we attach its label $\tilde{y}_i$ in $\tilde{y}_1^N$. Hence, we label te entire span and if the strategy results in two mentions where one contains the other, we eliminate the inner one.

Figure 1 displays the alignment between a Spanish sentence and its English automatic translation. It also shows a good match between the gold-standard tags in Spanish and the automatically extracted tags in English.

There are three ways in which we propose using these propagated labels:

1. Consider $\tilde{y}_1^N$ as the result of propagating the detected mentions in the original text x_1^N, basically selecting $y_1^N = \tilde{y}_1^N$. This situation corresponds to a case where no resources (annotated data) are available/needed on the source side, where the propagated labels are the output of the system.

2. Use the label sequence $\tilde{y}_1^N$ as an additional feature in the MaxEnt framework when predicting $P\left(y_j|x_1^N, y_{j-k}^{j-1}\right)$, together with other features built from resources available on the source language. We will call this model *CDP* (Context Dependent Propagation).

3. Starting with a large corpus (possibly including the training data), translate it into the resource-rich language and run mention detection. Then select the word sequences in the source language associated with the found mentions in the translation and add them to a machine-generated gazetteer $\mathcal{G}$[9]. This gazetteer $\mathcal{G}$ is then used to construct features for classification. We will call this model *CIP* (Context Independent Propagation).

From a runtime point of view, the CIP method has the advantage that there is no need to perform machine translation, and it can incorporate data from a very large amount of text. The CDP method, on the other hand, has the advantage that features are computed in context, and will not fire unless the corresponding mentions were found in the translated version (hence the name). Of course, the CDP method can incorporate features generated in the dictionary $\mathcal{G}$. The experimental section analyzes the impact of each of these techniques on mention detection task performance.

5 Resources

Experiments are conducted on the ACE 2007 data sets[10], in four languages: Arabic, Chinese, English, and Spanish. This data is selected from a variety of sources (broadcast news, broadcast conversations, newswire, web log, newswire, conversational telephony) and is labeled with 7 types: person, organization, location, facility, GPE (geo-political entity), vehicle and weapon. Besides mention level information, also labeled are coreference between the mentions, relations, events, and time resolution.

Since the evaluation tests set are not publicly available, we have split the publicly available *training* corpus into an 85%/15% data split. To facilitate future comparisons with work presented here, and to simulate a realistic scenario, the splits are created based on article dates: the test data is selected as the latest 15% of the data in chronological order, in each of the covered genres. This way, the documents in

[9]This is in fact a way to automatically construct a source-side mention dictionary.

[10]Same data as for ACE 2008.

Language	Training	Test
Arabic	323	56
Chinese	538	95
English	499	100
Spanish	467	52

Table 1: Datasets size (number of documents)

the training and test data sets do not overlap in time, and the content of the test data is more recent than the training data. Table 1 presents the number of documents in the training/test datasets for each of the four languages.

While performance on the ACE data is usually evaluated using a special-purpose measure - the ACE value metric (NIST, 2007), given that we are interested in the mention detection task only, we decided to use the more intuitive and popular (unweighted) F-measure, the harmonic mean of precision and recall.

6 Resource-Rich Languages

From the set of four languages in ACE 2007, we will unsurprisingly select English as the resource-rich language. Table 2 shows the performance of mention detection systems in all 4 languages one can obtain by using all available resources in that language, including lexical (words and morphs in a 3-word window, prefixes and suffixes of length up to 4, WordNet (Miller, 1995) for English), syntactic (POS tags, text chunks), and the output of other information extraction models.

	N	P	R	F
Arabic	3566	83.6	76.8	**80.0**
Chinese	4791	81.1	71.3	**75.8**
English	8170	84.6	80.8	**82.7**
Spanish	2487	79.1	73.5	**76.2**

Table 2: Performance of Arabic, Chinese, English and Spanish mention detection systems. Performance is presented in terms of Precision (P), Recall (R), and F-measure (F). The column (N) displays the number of mentions in the test set.

Results show that the English mention detection system has a better performance when compared to systems dealing with other languages such as Arabic, Chinese and Spanish. These results are not unexpected since the English model has access to a larger training data and uses richer set of information such as WordNet (Miller, 1995) and the output of a larger set of information extraction models.

Language Pair	BLEU Score
Arabic-English	0.55
Chinese-English	0.32
Spanish-English	0.55

Table 3: BLEU performance of the SMT systems on the 3 language pairs

7 Experiments

To show the effectiveness of cross-language mention propagation information in improving mention detection system performance in Arabic, Chinese and Spanish, we use three SMT systems with very competitive performance in terms of BLEU[11] (Papineni et al., 2002).

To give an idea of the SMT performance, Table 3 shows the performance of the translation systems on the three language pairs, computed on standard test sets. The Arabic to English SMT system is similar to the one described in (Huang and Papineni, 2007); it has 0.55 BLEU score on NIST 2003 Arabic-English machine translation evaluation test set. The Chinese to English SMT system has similar architecture to the one described in (Al-Onaizan and Papineni, 2006). This system obtains a score of 0.32 cased BLUE on NIST 2003 Arabic-English machine translation evaluation test set. The Spanish to English SMT system is similar to the one described in (Lee et al., 2006); it has a 0.55 BLEU score on the final text edition of the European Parliament Plenary Speech corpus in TC-STAR 2006 evaluation. As mentioned earlier, these three SMT systems have very competitive performance and are ranked among top 2 systems participating to NIST or TC-STAR evaluations. Also, the English mention detection system used for experiments has an F-measure of 82.7 and that has very competitive results among systems participating in the ACE 2007 evaluation.

Experiments are conducted under several conditions in order to investigate the effectiveness of our approach in improving mention detection system performance on languages with different levels of resource availability (from simple to more complex):

1. the system does not have access to any training data in the source language (no resources

[11]BLEU is an automatic measure for the translation quality which makes good use of multiple reference translations.

needed besides the MT system);

2. the system has access to only lexical information (information that can be directly derived exclusively from mention-labeled text);

3. the system has access to lexical and syntactic (e.g., POS tags, text chunks) information (requires mention-labeled text, and models to predict POS tags, etc);

4. the system that has access to lexical, syntactic, and semantic information (requires even more models and labeled data).

The rest of this section examines in detail these four cases.

To measure whether the improvement in performance of a particular system over another one is statistically significant or not, we use the stratified bootstrap re-sampling significance test (Noreen, 1989). This approach was used in the named entity recognition shared task of CoNNL-2002 (http://www.cnts.ua.ac.be/conll2002/ner/, 2002). In the following tables, we add a dagger sign † to results that are *not* statistically significant when compared to the baseline results.

7.1 No Source Language Training Data

In this first case, as described in Section 4, the mention labels in the source language are obtained directly through the alignment from the mentions in the translated text. This is a very simple scenario, which can be implemented with ease, and, as we will see, yields reasonable performance out-of-the-box.

	N	P	R	F
Arabic	3566	52.7	49.6	**51.1**
Chinese	4791	66.4	52.2	**58.5**
Spanish	2487	63.4	63.6	**63.5**

Table 4: Performance of the cross-language propagation from English mention detection system onto Arabic, Chinese and Spanish texts. Performance is presented in terms of Precision (P), Recall (R), and F-measure (F). The column (N) shows the number of mentions in the test set.

Experimental results presented in Table 4 show the performance of applying this information transfer approach. For each source language (Arabic, Chinese, or Arabic), we show the performance of propagating mentions from the English text. Even though no training data to build a source language mention classifier is available, we still can detect mentions with reasonably high accuracy. We consider the obtained accuracy as reasonably good because, as an example, the performance of a system that attaches to every word its most frequent label (unigram) is around 25% F-measure on Arabic. Results in Table 4 also show that even though the Chinese-to-English SMT system is lower in term of BLEU than the Arbic-to-English SMT system (0.32 vs. 0.55), performance of the cross-language propagation from English mention detection system onto Chinese is better than the performance of the propagation from English mention detection system onto Arabic. One reason for this is that we notice that Chinese-to-English SMT system translates and aligns ACE categories better than Arabic-to-English SMT system.

7.2 Lexical Resources

In this section, we consider the case when we have available training data in the source language to be able to train a statistical classifier. We also consider that the classifier has access to lexical information *only*. Our goal here is to study the effectiveness of *adding* cross-language mention propagation information to improve mention detection performance on languages with limited resources.

Table 5 shows the performance of the 3 languages with and without cross-language mention propagation information from English, with the 3 propagation methods described in Section 4. One can see that propagating mention propagation information results in system performance increase[12]. When systems use the CIP method, no improvement can be observed on Arabic and Chinese, while a small improvement of 0.5F point is obtained on Spanish (74.5 vs. 75.0). In contrast, when systems use the CDP method an improvement is obtained in recall – which is to be expected, given the method – leading to systems with better performance in terms of F-measure: 1.6F points improvement for Arabic, 1.5F points improvement for Chinese and almost 3F points improvement for Spanish. The results for all the CDP transfers and the CIP for Spanish are statistically significant.

7.3 Lexical and Syntactic Resources

We represent in Table 6 mention detection system performance when syntactic resources are available in the source language, in addition to lexical re-

[12]Only systems' performance marked with † is not statistically significantly better.

		Baseline			CIP			CDP		
	N	P	R	F	P	R	F	P	R	F
Arabic:	3566	81.8	71.7	**76.4**	82.2	71.3	**76.4**†	82.6	73.9	**78.0**
Chinese:	4791	79.3	70.2	**74.5**	79.4	70.5	**74.7**†	79.8	72.5	**76.0**
Spanish:	2478	79.1	70.4	**74.5**	79.7	70.8	**75.0**	80.4	74.6	**77.4**

Table 5: Performance of Arabic, Chinese and Spanish mention detection using lexical features ("Baseline" column). Columns "CIP" stands for systems that add cross-language context independent mention propagation information and column "CDP" is for systems that add cross-language context dependent mention propagation information.

		Baseline			CIP			CDP		
	N	P	R	F	P	R	F	P	R	F
Arabic:	3566	82.2	72.6	**77.1**	82.7	72.9	**77.5**	83.2	74.5	**78.6**
Chinese:	4791	80.0	71.3	**75.5**	79.9	71.5	**75.5**†	81.0	72.4	**76.5**
Spanish:	2487	79.1	71.2	**74.9**	79.9	71.9	**75.7**	80.7	74.6	**77.5**

Table 6: Performance of Arabic, Chinese and Spanish mention detection using lexical and syntactic features (POS tags, chunk information, etc).

sources available in the previous Subsection. This experiment is important because it tests the effectiveness of the propagation approach in improving performance on languages with a typical level of resources.

Results show that even in this situation, the use of cross language mention propagation information still lead to considerable improvement: using the CDP transfer method yields improvements from $1.1F$ in Chinese to $2.6F$ in Spanish. Similar to the previous section, the use of CIP information did not improve performance significantly on Arabic (77.5 vs. 77.1) and Chinese (75.5 vs. 75.5) systems, but we notice an improvement in Spanish[13].

7.4 Lexical, Syntactic and Semantic Resources

This final section investigates whether the access to cross-language mention propagation information can still improve the performance of existing competitive mention detection systems trained on languages with large resources. In this case, systems have access to a full array of lexical, syntax, semantic information, including the output from other information extraction models. Table 7 presents the performance of mention detection systems on the three languages, in the familiar 3 propagation methods: again, results show that better performance is obtained when cross language mention information is used. Under CIP, almost no change in terms of performance is obtained for Arabic and Spanish, though a slight improvement can be observed for Chinese (76.9F vs. 75.8F). When CDP is used the performance of mention detection systems is improved by 0.9F for Arabic (80.9 vs. 80.0), 2.3F for Chinese (78.1F vs. 75.8F) and 1.9F for Spanish (78.1 vs. 76.2F). Once again, the results prove that the use of cross language mention propagation information, especially through CDP, is effective in improving the performance even in this case.

By comparing results across tables, one can note that systems having access to only lexical and cross language mention propagation information are as effective as systems having access to large set of information. For Chinese, we obtain a performance of 75.8F when the system has access to lexical, syntactic and output of other information extraction models. On the other hand, the same system has a slightly better performance of 76.0 when it has access to lexical and cross language mention propagation information. The same behavior is observed for Spanish, we obtain a performance of 76.2F when the system has access to lexical, syntactic and output of other information extraction models; compared to 77.4F when lexical and cross language mention information are used. This is not true for Arabic where having access to larger set of information led to better performance when compared to systems having access to lexical information and CDP information (80.0F vs. 78.0). We attribute this difference to the fact that in Arabic we use the output of larger number of information extraction models, and consequently a richer set of information.

[13]The dagger sign † marks the systems that are not statistically significantly better.

		Baseline			CIP			CDP		
	N	P	R	F	P	R	F	P	R	F
Arabic:	3566	83.6	76.8	**80.0**	83.9	77.0	**80.2**†	84.2	77.8	**80.9**
Chinese:	4791	81.1	71.3	**75.8**	81.4	73.0	**76.9**	81.7	74.8	**78.1**
Spanish:	2487	79.1	73.5	**76.2**	79.3	73.4	**76.2**†	80.1	76.2	**78.1**

Table 7: Performance of Arabic, Chinese and Spanish mention detection using lexical, syntactic and output of other information extraction models: full-blown systems.

The other observation that is worth making is that the improvement in performance has a decreasing tendency as more resources are available. The performance gain for CDP in Arabic goes from 1.6 to 1.5 to 0.9, and the one on Spanish goes from 2.9 to 2.6 to 1.9. The one on Chinese follows part of this trend, as it goes from 1.4 to 1.1 to 2.3. While the evidence here is not definitive, one can indeed note the reduced effectiveness of the method as more resources are available, which was indeed what we expected.

Results obtained by all these experiments help answer an important question: when trying to improve mention detection systems in a resource-poor language, should we invest in building resources or should we use propagation from a resource-rich language to (at least) bootstrap the process? The answer seems to be the latter.

8 Conclusion

This paper presents a new approach to mention detection in low, medium or high-resource languages, which benefits from projecting the output from a resource-rich language such as English. We show that even when no training data is available in one source language, we can still build a decently performing baseline mention detection system by only using resources from English. This approach requires a mention detection system on a resource-rich language and an SMT system that translate text from the source to the resource-rich language, both of which can be attained.

In cases when large resources are available in the source language, our cross language mention propagation technique is still able to further improve mention detection system performance. Experiments performed on the four languages of ACE 2007, with English chosen as the *resource-rich* language, show consistent and significant improvements across conditions and levels of linguistic sophistication. The experiments are conducted on clearly specified partitions of the ACE 2007 data set, so future comparisons against the presented work can be correctly and accurately made. We also note that systems that have access to lexical and cross language mention propagation information are as accurate as those that have access to lexical, syntactic and output of other information extraction models in the source language (but no cross-language resources). As future work, we plan to extend this work to use semi-supervised and unsupervised approaches that can make use of cross-language information propagation.

We believe that it is important for the research community to continue to invest in building better resources in “source” languages, as it looks the most promising approach. However, using a propagation approach can definitely help bootstrap the process.

Acknowledgments

This work was supported by DARPA/IPTO Contract No. HR0011-06-2-0001 under the GALE program.

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 Association for Computational Linguistics*, pages 529–536, Sydney, Australia, July. Association for Computational Linguistics.

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

C. Cabezas, B. Dorr, and P. Resnik. 2001. Spanish language processing at university of maryland: Building infrastructure for multilingual applications. In *Proceedings of the 2nd International Workshop on Spanish Language Processing and Language Technologies*.

Stanley Chen and Ronald Rosenfeld. 2000. A survey of smoothing techniques for me models. *IEEE Trans. on Speech and Audio Processing*.

I. Dagan and A. Itai. 1994. Word sense disambiguation using a second language monolingual corpus. *Computational Linguistics*, 20(4):563–596.

Ido Dagan, Alon Itai, and Ulrike Schwall. 1991. Two languages are more informative than one. In *Meeting of the Association for Computational Linguistics*, pages 130–137.

Mona Diab and Philip Resnik. 2001. An unsupervised method for word sense tagging using parallel corpora. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 255–262, Morristown, NJ, USA. Association for Computational Linguistics.

R. Florian, H. Hassan, A. Ittycheriah, H. Jing, N. Kambhatla, X. Luo, N Nicolov, and S Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 1–8.

W. Gale, K. Church, and D. Yarowsky. 1992. A method for disambiguating word senses in a large corpus. *Computers and the Humanities*, 26:415–439.

Joshua Goodman. 2002. Sequential conditional generalized iterative scaling. In *Proceedings of ACL'02*.

Gregory Grefenstette. 1998. *Cross-Language Information Retrieval*, volume 079238122X. Kluwer Academic Publishers.

http://www.cnts.ua.ac.be/conll2002/ner/. 2002.

Fei Huang and Kishore Papineni. 2007. Hierarchical system combination for machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 277–286.

Rebecca Hwa, Philip Resnik, and Amy Weinberg. 2002. Breaking the resource bottleneck for multilingual parsing. In *Proceedings of the Workshop on Linguistic Knowledge Acquisition and Representation: Bootstrapping Annotated Language Data*.

H. Jing, R. Florian, X. Luo, T. Zhang, and A. Ittycheriah. 2003. HowtogetaChineseName(Entity): Segmentation and combination issues. In *Proceedings of EMNLP'03*, pages 200–207.

Philipp Koehn. 2004. Pharaoh: a Beam Search Decoder for Phrase-Based Statistical Machine Translation Models. In *Proceedings of AMTA'04*, Washington DC, September-October.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Y.-S. Lee, K. Papineni, S. Roukos, O. Emam, and H. Hassan. 2003. Language model based Arabic word segmentation. In *Proceedings of the ACL'03*, pages 399–406.

Young-Suk Lee, Yaser Al-Onaizan, Kishore Papineni, and Salim Roukos. 2006. Ibm spoken language translation system. In *TC-STAR Workshop on Speech-to-Speech Translation*, pages 13–18, Barcelona, Spain, June.

Andrew McCallum, Dayne Freitag, and Fernando Pereira. 2000. Maximum entropy markov models for information extraction and segmentation. In *ICML*.

G. A. Miller. 1995. WordNet: A lexical database. *Communications of the ACM*, 38(11).

NIST. 2007. The ACE evaluation plan. www.nist.gov/speech/tests/ace/index.htm.

Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses*. John Wiley Sons.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

P.F.Brown, S.A.Della Pietra, V.J. Della Pietra, and R.L.Mercer. 1991. Word-sense disambiguation using statistical methods. In *Proceedings of ACL'91*.

L. Ramshaw and M. Marcus. 1994. Exploring the statistical derivation of transformational rule sequences for part-of-speech tagging. In *Proceedings of the ACL Workshop on Combining Symbolic and Statistical Approaches to Language*, pages 128–135.

L. Ramshaw and M. Marcus. 1995. Text chunking using transformation-based learning. In David Yarowsky and Kenneth Church, editors, *Proceedings of the Third Workshop on Very Large Corpora*, pages 82–94, Somerset, New Jersey. Association for Computational Linguistics.

E. Riloff, C. Schafer, and D. Yarowsky. 2002. Inducing information extraction systems for new languages via cross-language projection. In *Proceedings of Coling 2002*, Taipei, Taiwan.

E. F. Tjong Kim Sang and J. Veenstra. 1999. Representing text chunks. In *Proceedings of EACL'99*.

E. F. Tjong Kim Sang. 2002. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2002*, pages 155–158.

D. Yarowsky, G. Ngai, and R. Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of HLT 2001*, San Diego, California, USA.

Imed Zitouni, Jeff Sorensen, Xiaoqiang Luo, and Radu Florian. 2005. The impact of morphological stemming on Arabic mention detection and coreference resolution. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 63–70, Ann Arbor, June.

Decomposability of Translation Metrics for Improved Evaluation and Efficient Algorithms

David Chiang and **Steve DeNeefe**
Information Sciences Institute
University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292 USA
{chiang,sdeneefe}@isi.edu

Yee Seng Chan and **Hwee Tou Ng**
Department of Computer Science
National University of Singapore
Law Link
Singapore 117590
{chanys,nght}@comp.nus.edu.sg

Abstract

Bleu is the *de facto* standard for evaluation and development of statistical machine translation systems. We describe three real-world situations involving comparisons between different versions of the same systems where one can obtain improvements in Bleu scores that are questionable or even absurd. These situations arise because Bleu lacks the property of *decomposability*, a property which is also computationally convenient for various applications. We propose a very conservative modification to Bleu and a cross between Bleu and word error rate that address these issues while improving correlation with human judgments.

1 Introduction

Bleu (Papineni et al., 2002) was one of the first automatic evaluation metrics for machine translation (MT), and despite being challenged by a number of alternative metrics (Melamed et al., 2003; Banerjee and Lavie, 2005; Snover et al., 2006; Chan and Ng, 2008), it remains the standard in the statistical MT literature. Callison-Burch et al. (2006) have subjected Bleu to a searching criticism, with two real-world case studies of significant failures of correlation between Bleu and human adequacy/fluency judgments. Both cases involve comparisons between statistical MT systems and other translation methods (human post-editing and a rule-based MT system), and they recommend that the use of Bleu be restricted to comparisons between related systems or different versions of the same systems. In Bleu's defense, comparisons between different versions of the same system were exactly what Bleu was designed for.

However, we show that even in such situations, difficulties with Bleu can arise. We illustrate three ways that properties of Bleu can be exploited to yield improvements that are questionable or even absurd. All of these scenarios arose in actual practice and involve comparisons between different versions of the same statistical MT systems. They can be traced to the fact that Bleu is not *decomposable* at the sentence level: that is, it lacks the property that improving a sentence in a test set leads to an increase in overall score, and degrading a sentence leads to a decrease in the overall score. This property is not only intuitive, but also computationally convenient for various applications such as translation reranking and discriminative training. We propose a minimal modification to Bleu that reduces its nondecomposability, as well as a cross between Bleu and word error rate (WER) that is decomposable down to the subsentential level (in a sense to be made more precise below). Both metrics correct the observed problems and correlate with human judgments better than Bleu.

2 The Bleu metric

Let $g_k(w)$ be the multiset of all k-grams of a sentence w. We are given a sequence of candidate translations $\mathbf{c}$ to be scored against a set of sequences of reference translations, $\{\mathbf{r}^j\} = \mathbf{r}^1, \ldots, \mathbf{r}^R$:

$$\begin{aligned}
\mathbf{c} &= c_1, c_2, c_3, \ldots, c_N \\
\mathbf{r}^1 &= r^1_1, r^1_2, r^1_3, \ldots, r^1_N \\
&\vdots \\
\mathbf{r}^R &= r^R_1, r^R_2, r^R_3, \ldots, r^R_N
\end{aligned}$$

Then the Bleu score of **c** is defined to be

$$\text{Bleu}(\mathbf{c}, \{\mathbf{r}^j\}) = \prod_{k=1}^{4} pr_k(\mathbf{c}, \{\mathbf{r}^j\})^{\frac{1}{4}} \times bp(\mathbf{c}, \{\mathbf{r}^j\}) \quad (1)$$

where[1]

$$pr_k(\mathbf{c}, \{\mathbf{r}^j\}) = \frac{\sum_i \left| g_k(c_i) \cap \bigcup_j g_k(r_i^j) \right|}{\sum_i |g_k(c_i)|} \quad (2)$$

is the k-gram precision of **c** with respect to $\{\mathbf{r}^j\}$, and $bp(\mathbf{c}, \mathbf{r})$, known as the *brevity penalty*, is defined as follows. Let $\phi(x) = \exp(1 - 1/x)$. In the case of a single reference **r**,

$$bp(\mathbf{c}, \mathbf{r}) = \phi\left(\min\left\{1, \frac{\sum_i |c_i|}{\sum_i |r_i|}\right\}\right) \quad (3)$$

In the multiple-reference case, the length $|r_i|$ is replaced with an *effective reference length*, which can be calculated in several ways.

- In the original definition (Papineni et al., 2002), it is the length of the reference sentence whose length is *closest* to the test sentence.
- In the NIST definition, it is the length of the *shortest* reference sentence.
- A third possibility would be to take the *average* length of the reference sentences.

The purpose of the brevity penalty is to prevent a system from generating very short but precise translations, and the definition of effective reference length impacts how strong the penalty is. The NIST definition is the most tolerant of short translations and becomes more tolerant with more reference sentences. The original definition is less tolerant but has the counterintuitive property that decreasing the length of a test sentence can eliminate the brevity penalty. Using the average reference length seems attractive but has the counterintuitive property that an exact match with one of the references may not get a 100% score. Throughout this paper we use the NIST definition, as it is currently the definition most used in the literature and in evaluations.

The brevity penalty can also be seen as a stand-in for recall. The fraction $\frac{\sum_i |c_i|}{\sum_i |r_i|}$ in the definition of the brevity penalty (3) indeed resembles a weak recall score in which every guessed item counts as a match. However, with recall, the per-sentence score $\frac{|c_i|}{|r_i|}$ would never exceed unity, but with the brevity penalty, it can. This means that if a system generates a long translation for one sentence, it can generate a short translation for another sentence without facing a penalty. This is a serious weakness in the Bleu metric, as we demonstrate below using three scenarios, encountered in actual practice.

3 Exploiting the Bleu metric

3.1 The sign test

We are aware of two methods that have been proposed for significance testing with Bleu: bootstrap resampling (Koehn, 2004b; Zhang et al., 2004) and the sign test (Collins et al., 2005). In bootstrap resampling, we sample with replacement from the test set to synthesize a large number of test sets, and then we compare the performance of two systems on those synthetic test sets to see whether one is better 95% (or 99%) of the time. But Collins et al. (2005) note that it is not clear whether the conditions required by bootstrap resampling are met in the case of Bleu, and recommend the sign test instead. Suppose we want to determine whether a set of outputs **c** from a test system is better or worse than a set of baseline outputs **b**. The sign test requires a function $f(b_i, c_i)$ that indicates whether c_i is a better, worse, or same-quality translation relative to b_i. However, because Bleu is not defined on single sentences, Collins et al. use an approximation: for each i, form a composite set of outputs $\mathbf{b}' = \{b_1, \ldots, b_{i-1}, c_i, b_{i+1}, \ldots, b_N\}$, and compare the Bleu scores of **b** and $\mathbf{b}'$.

The goodness of this approximation depends on to what extent the comparison between **b** and $\mathbf{b}'$ is dependent only on b_i and c_i, and independent of the other sentences. However, Bleu scores are highly context-dependent: for example, if the sentences in **b** are on average ϵ words longer than the reference sentences, then c_i can be as short as $(N-1)\epsilon$ words

[1]We use the following definitions about multisets: if X is a multiset, let $\#_X(a)$ be the number of times a occurs in X. Then:

$$|X| \equiv \sum_a \#_X(a)$$
$$\#_{X\cap Y}(a) \equiv \min\{\#_X(a), \#_Y(a)\}$$
$$\#_{X\cup Y}(a) \equiv \max\{\#_X(a), \#_Y(a)\}$$

shorter than r_i without incurring the brevity penalty. Moreover, since the c_i are substituted in one at a time, we can do this for all of the c_i. Hence, $\mathbf{c}$ could have a disastrously low BLEU score (because of the brevity penalty) yet be found by the sign test to be significantly better than the baseline.

We have encountered this situation in practice: two versions of the same system with BLEU scores of 29.6 (length ratio 1.02) and 29.3 (length ratio 0.97), where the sign test finds the second system to be significantly better than the first (and the first system significantly better than the second). Clearly, in order for a significance test to be sensible, it should not contradict the observed scores, and should certainly not contradict itself. In the rest of this paper, except where indicated, all significance tests are performed using bootstrap resampling.

3.2 Genre-specific training

For several years, much statistical MT research has focused on translating newswire documents. One likely reason is that the DARPA TIDES program used newswire documents for evaluation for several years. But more recent evaluations have included other genres such as weblogs and conversation. The conventional wisdom has been that if one uses a single statistical translation system to translate text from several different genres, it may perform poorly, and it is better to use several systems optimized separately for each genre.

However, if our task is to translate documents from multiple known genres, but they are evaluated together, the BLEU metric allows us to use that fact to our advantage. To understand how, notice that our system has an optimal number of words that it should generate for the entire corpus: too few and it will be penalized by BLEU's brevity penalty, and too many increases the risk of additional non-matching k-grams. But these words can be distributed among the sentences (and genres) in any way we like. Instead of translating sentences from each genre with the best genre-specific systems possible, we can generate longer outputs for the genre we have more confidence in, while generating shorter outputs for the harder genre. This strategy will have mediocre performance on each individual genre (according to both intuition and BLEU), yet will receive a higher BLEU score on the combined test set than the combined systems optimized for each genre.

In fact, knowing which sentence is in which genre is not even always necessary. In one recent task, we translated documents from two different genres, without knowing the genre of any given sentence. The easier genre, newswire, also tended to have shorter reference sentences (relative to the source sentences) than the harder genre, weblogs. For example, in one dataset, the newswire reference sets had between 1.3 and 1.37 English words per Arabic word, but the weblog reference set had 1.52 English words per Arabic word. Thus, a system that is uniformly verbose across both genres will apportion more of its output to newswire than to weblogs, serendipitously leading to a higher score. This phenomenon has subsequently been observed by Och (2008) as well.

We trained three Arabic-English syntax-based statistical MT systems (Galley et al., 2004; Galley et al., 2006) using max-BLEU training (Och, 2003): one on a newswire development set, one on a weblog development set, and one on a combined development set containing documents from both genres. We then translated a new mixed-genre test set in two ways: (1) each document with its appropriate genre-specific system, and (2) all documents with the system trained on the combined (mixed-genre) development set. In Table 3, we report the results of both approaches on the entire test dataset as well as the portion of the test dataset in each genre, for both the genre-specific and mixed-genre trainings.

The genre-specific systems each outperform the mixed system on their own genre as expected, but when the same results are combined, the mixed system's output is a full BLEU point higher than the combination of the genre-specific systems. This is because the mixed system produces outputs that have about 1.35 English words per Arabic word on average: longer than the shortest newswire references, but shorter than the weblog references. The mixed system does worse on each genre but better on the combined test set, whereas, according to intuition, a system that does worse on the two subsets should also do worse on the combined test set.

3.3 Word deletion

A third way to take advantage of the BLEU metric is to permit an MT system to delete arbitrary words in the input sentence. We can do this by introducing new phrases or rules into the system that match words in the input sentence but generate no output; to these rules we attach a feature whose weight is tuned during max-Bleu training. Such rules have been in use for some time but were only recently discussed by Li et al. (2008).

When we add word-deletion rules to our MT system, we find that the Bleu increases significantly (Table 6, line 2). Figure 1 shows some examples of deletion in Chinese-English translation. The first sentence has a proper name, 麦格赛赛/*maigesaisai* 'Magsaysay', which has been mistokenized into four tokens. The baseline system attempts to translate the first two phonetic characters as "wheat Georgia," whereas the other system simply deletes them. On the other hand, the second sentence shows how word deletion can sacrifice adequacy for the sake of fluency, and the third sentence shows that sometimes word deletion removes words that could have been translated well (as seen in the baseline translation).

Does Bleu reward word deletion fairly? We note two reasons why word deletion might be desirable. First, some function words should truly be deleted: for example, the Chinese particle 的/*de* and Chinese measure words often have no counterpart in English (Li et al., 2008). Second, even content word deletion might be helpful if it allows a more fluent translation to be assembled from the remnants. We observe that in the above experiment, word deletion caused the absolute number of k-gram matches, and not just k-gram precision, to increase for all $1 \leq k \leq 4$.

Human evaluation is needed to conclusively determine whether Bleu rewards deletion fairly. But to control for these potentially positive effects of deletion, we tested a *sentence-deletion* system, which is the same as the word-deletion system but constrained to delete *all* of the words in a sentence or none of them. This system (Table 6, line 3) deleted 8–10% of its input and yielded a Bleu score with no significant decrease ($p \geq 0.05$) from the baseline system's. Given that our model treats sentences independently, so that it cannot move information from one sentence to another, we claim that deletion of nearly 10% of the input is a grave translation deficiency, yet Bleu is insensitive to it.

What does this tell us about word deletion? While acknowledging that some word deletions can improve translation quality, we suggest in addition that because word deletion provides a way for the system to translate the test set selectively, a behavior which we have shown that Bleu is insensitive to, part of the score increase due to word deletion is likely an artifact of Bleu.

4 Other metrics

Are other metrics susceptible to the same problems as the Bleu metric? In this section we examine several other popular metrics for these problems, propose two of our own, and discuss some desirable characteristics for any new MT evaluation metric.

4.1 Previous metrics

We ran a suite of other metrics on the above problem cases to see whether they were affected. In none of these cases did we repeat minimum-error-rate training; all these systems were trained using max-Bleu. The metrics we tested were:

- METEOR (Banerjee and Lavie, 2005), version 0.6, using the exact, Porter-stemmer, and WordNet synonmy stages, and the optimized parameters $\alpha = 0.81, \beta = 0.83, \gamma = 0.28$ as reported in (Lavie and Agarwal, 2007).

- GTM (Melamed et al., 2003), version 1.4, with default settings, except $e = 1.2$, following the WMT 2007 shared task (Callison-Burch et al., 2007).

- MaxSim (Chan and Ng, 2008), more specifically MaxSim$_n$, which skips the dependency relations.

On the sign test (Table 2), all metrics found significant differences consistent with the difference in score between the two systems. The problem related to genre-specific training does not seem to affect the other metrics (see Table 4), but they still manifest the unintuitive result that genre-specific training is sometimes worse than mixed-genre training. Finally, all metrics but GTM disfavored both word deletion and sentence deletion (Table 7).

4.2 Strict brevity penalty

A very conservative way of modifying the Bleu metric to combat the effects described above is to im-

(a)	source	费 孝通 被 授予 麦 格 赛 赛 奖
	reference	fei xiaotong awarded magsaysay prize
	baseline	fei xiaotong was awarded the wheat georgia xaixai prize
	delete	fei xiaotong was awarded xaixai award
(b)	source	雨 花 石 正中 是 一 幅 十分 清晰 的 中华人民共和国 版图 的 图象 。
	reference	the center of the yuhua stone bears an image which very much resembles the territory of the people 's republic of china .
	baseline	rain huashi center is a big clear images of chinese territory .
	delete	rain is a clear picture of the people 's republic of china .
(c)	source	城建 成为 外商 投资 青海 新 热点
	reference	urban construction becomes new hotspot for foreign investment in qinghai
	baseline	urban construction become new hotspot for foreign investment qinghai
	delete	become new foreign investment hotspot

Figure 1: Examples of word deletion. Underlined Chinese words were deleted in the word-deletion system; underlined English words correspond to deleted Chinese words.

pose a stricter brevity penalty. In Section 2, we presented the brevity penalty as a stand-in for recall, but noted that unlike recall, the per-sentence score $\frac{|c_i|}{|r_i|}$ can exceed unity. This suggests the simple fix of clipping the per-sentence recall scores in a similar fashion to the clipping of precision scores:

$$bp(\mathbf{c}, \mathbf{r}) = \phi\left(\frac{\sum_i \min\{|c_i|, |r_i|\}}{\sum_i |r_i|}\right) \quad (4)$$

Then if a translation system produces overlong translations for some sentences, it cannot use those translations to license short translations for other sentences. Call this revised metric Bleu-sbp (for Bleu *with strict brevity penalty*).

We can test this revised definition on the problem cases described above. Table 2 shows that Bleu-sbp resolves the inconsistency observed between Bleu and the sign test, using the example test sets from Section 3.1 (no max-Bleu-sbp training was performed). Table 5 shows the new scores of the mixed-genre example from Section 3.2 after max-Bleu-sbp training. These results fall in line with intuition—tuning separately for each genre leads to slightly better scores in all cases. Finally, Table 8 shows the Bleu-sbp scores for the word-deletion example from Section 3.3, using both max-Bleu training and max-Bleu-sbp training. We see that Bleu-sbp reduces the benefit of word deletion to an insignificant level on the test set, and severely punishes sentence deletion. When we retrain using max-Bleu-sbp, the rate of word deletion is reduced and sentence deletion is all but eliminated, and there are no significant differences on the test set.

4.3 4-gram recognition rate

All of the problems we have examined—except for word deletion—are traceable to the fact that Bleu is not a sentence-level metric. Any metric which is defined as a weighted average of sentence-level scores, where the weights are system-independent, will be immune to these problems. Note that any metric involving micro-averaged precision (in which the sentence-level counts of matches and guesses are summed separately before forming their ratio) cannot have this property. Of the metrics surveyed in the WMT 2007 evaluation-evaluation (Callison-Burch et al., 2007), at least the following metrics have this property: WER (Nießen et al., 2000), TER (Snover et al., 2006), and ParaEval-Recall (Zhou et al., 2006).

Moreover, this evaluation concern dovetails with a frequent engineering concern, that sentence-level scores are useful at various points in the MT pipeline: for example, minimum Bayes risk decoding (Kumar and Byrne, 2004), selecting oracle translations for discriminative reranking (Liang et al., 2006; Watanabe et al., 2007), and sentence-by-sentence comparisons of outputs during error analysis. A variation on Bleu is often used for these purposes, in which the k-gram precisions are "smoothed" by adding one to the numerator and denominator (Lin and Och, 2004); this addresses the problem of a zero k-gram match canceling out the entire score, but it does not address the problems illustrated above.

The remaining issue, word deletion, is more difficult to assess. It could be argued that part of the gain due to word deletion is caused by Bleu allowing a system to selectively translate those *parts* of a sentence on which higher precision can be obtained. It would be difficult indeed to argue that an evaluation metric, in order to be fair, must be decomposable into subsentential scores, and we make no such claim. However, there is again a dovetailing engineering concern which is quite legitimate. If one wants to select the minimum-Bayes-risk translation from a *lattice* (or shared forest) instead of an n-best list (Tromble et al., 2008), or to select an oracle translation from a lattice (Tillmann and Zhang, 2006; Dreyer et al., 2007; Leusch et al., 2008), or to perform discriminative training on all the examples contained in a lattice (Taskar et al., 2004), one would need a metric that can be calculated on the edges of the lattice.

Of the metrics surveyed in the WMT 2007 evaluation-evaluation, only one metric, to our knowledge, has this property: word error rate (Nießen et al., 2000). Here, we deal with the related *word recognition rate* (McCowan et al., 2005),

$$\begin{aligned}\mathrm{WRR} &= 1 - \mathrm{WER}\\ &= 1 - \min \frac{I + D + S}{|r|}\\ &= \max \frac{M - I}{|r|}\end{aligned} \tag{5}$$

where I is the number of insertions, D of deletions, S of substitutions, and $M = |r| - D - S$ the number of matches. The dynamic program for WRR can be formulated as a Viterbi search through a finite-state automaton: given a candidate sentence c and a reference sentence r, find the highest-scoring path matching c through the automaton with states $0, \ldots, |r|$, initial state 0, final state $|r|$, and the following transitions (a $\star$ matches any symbol):

For $0 \leq i < |r|$:

$$i \xrightarrow{r_{i+1}:1} i + 1 \qquad \text{match}$$

$$i \xrightarrow{\epsilon:0} i + 1 \qquad \text{deletion}$$

$$i \xrightarrow{\star:0} i + 1 \qquad \text{substitution}$$

For $0 \leq i \leq |r|$:

$$i \xrightarrow{\star:-1} i \qquad \text{insertion}$$

This automaton can be intersected with a typical stack-based phrase-based decoder lattice (Koehn, 2004a) or CKY-style shared forest (Chiang, 2007) in much the same way that a language model can, yielding a polynomial-time algorithm for extracting the best-scoring translation from a lattice or forest (Wagner, 1974). Intuitively, the reason for this is that WRR, like most metrics, implicitly constructs a word alignment between c and r and only counts matches between aligned words; but unlike other metrics, this alignment is constrained to be monotone.

We can combine WRR with the idea of k-gram matching in Bleu to yield a new metric, the *4-gram recognition rate*:

$$\text{4-GRR} = \max \frac{\sum_{k=1}^{4} M_k - \alpha I - \beta D}{\sum_{k=1}^{4} |g_k(r)|} \tag{6}$$

where M_k is the number of k-gram matches, α and β control the penalty for insertions and deletions, and g_k is as defined in Section 2. We presently set $\alpha = 1, \beta = 0$ by analogy with WRR, but explore other settings below. To calculate 4-GRR on a whole test set, we sum the numerators and denominators as in micro-averaged recall.

The 4-GRR can also be formulated as a finite-state automaton, with states $\{(i, m) \mid 0 \leq i \leq |r|, 0 \leq m \leq 3\}$, initial state $(0, 0)$, final states $(|r|, m)$, and the following transitions:

For $0 \leq i < |r|$, $0 \leq m \leq 3$:

$$(i, m) \xrightarrow{r_{i+1}:m+1} (i + 1, \min\{m + 1, 3\}) \qquad \text{match}$$

$$(i, m) \xrightarrow{\epsilon:-\beta} (i + 1, 0) \qquad \text{deletion}$$

$$(i, m) \xrightarrow{\star:0} (i + 1, 0) \qquad \text{substitution}$$

Metric	Adq	Flu	Rank	Con	Avg
Sem. role overlap	77.4	83.9	80.3	74.1	78.9
ParaEval recall	71.2	74.2	76.8	79.8	75.5
METEOR	70.1	71.9	74.5	66.9	70.9
Bleu	68.9	72.1	67.2	60.2	67.1
WER	51.0	54.2	34.5	52.4	48.0
Bleu-sbp	73.9	76.7	73.5	63.4	71.9
4-GRR	72.3	75.5	74.3	64.2	71.6

Table 1: Our new metrics correlate with human judgments better than Bleu (case-sensitive). Adq = Adequacy, Flu = Fluency, Con = Constituent, Avg = Average.

For $0 \leq i \leq |r|$, $0 \leq m \leq 3$:

$$(i, m) \xrightarrow{\star:-\alpha} (i, 0) \qquad \text{insertion}$$

Therefore 4-GRR can also be calculated efficiently on lattices or shared forests.

We did not attempt max-4-GRR training, but we evaluated the word-deletion test sets obtained by max-Bleu and max-Bleu-sbp training using 4-GRR. The results are shown in Table 7. In general, the results are very similar to Bleu-sbp except that 4-GRR sometimes scores word deletion slightly lower than baseline.

5 Correlation with human judgments

The shared task of the 2007 Workshop on Statistical Machine Translation (Callison-Burch et al., 2007) was conducted with several aims, one of which was to measure the correlation of several automatic MT evaluation metrics (including Bleu) against human judgments. The task included two datasets (one drawn from the Europarl corpus and the other from the News Commentary corpus) and across three language pairs (from German, Spanish, and French to English, and back). In our experiments, we focus on the tasks where the target language is English.

For human evaluations of the MT submissions, four different criteria were used:

- Adequacy: how much of the meaning expressed in the reference translation is also expressed in the hypothesis translation.
- Fluency: how well the translation reads in the target language.
- Rank: each translation is ranked from best to worst, relative to the other translations of the same sentence.
- Constituent: constituents are selected from source-side parse-trees, and human judges are asked to rank their translations.

We scored the workshop shared task submissions with Bleu-sbp and 4-GRR, then converted the raw scores to rankings and calculated the Spearman correlations with the human judgments. Table 1 shows the results along with Bleu and the three metrics that achieved higher correlations than Bleu: semantic role overlap (Giménez and Márquez, 2007), ParaEval recall (Zhou et al., 2006), and METEOR (Banerjee and Lavie, 2005). We find that both our proposed metrics correlate with human judgments better than Bleu does.

However, recall the parameters α and β in the definition of 4-GRR that control the penalty for inserted and deleted words. Experimenting with this parameter reveals that $\alpha = -0.9, \beta = 1$ yields a correlation of 78.9%. In other words, a metric that unboundedly rewards spuriously inserted words correlates better with human judgments than a metric that punishes them. We assume this is because there are not enough data points (systems) in the sample and ask that all these figures be taken with a grain of salt. As a general remark, it may be beneficial for human-correlation datasets to include a few straw-man systems that have very short or very long translations.

6 Conclusion

We have described three real-world scenarios involving comparisons between different versions of the same statistical MT systems where Bleu gives counterintuitive results. All these issues center around the issue of decomposability: the sign test fails because substituting translations one sentence at a time can improve the overall score yet substituting them all at once can decrease it; genre-specific training fails because improving the score of two halves of a test set can decrease the overall score; and sentence deletion is not harmful because generating empty translations for selected sentences does not necessarily decrease the overall score.

We proposed a minimal modification to Bleu, called Bleu-sbp, and showed that it ameliorates these problems. We also proposed a metric, 4-GRR, that is decomposable at the sentence level and is therefore guaranteed to solve the sign test, genre-specific tuning, and sentence deletion problems; moreoever, it is decomposable at the subsentential level, which has potential implications for evaluating word deletion and promising applications to translation reranking and discriminative training.

Acknowledgments

Our thanks go to Daniel Marcu for suggesting modifying the Bleu brevity penalty, and to Jonathan May and Kevin Knight for their insightful comments. This research was supported in part by DARPA grant HR0011-06-C-0022 under BBN Technologies subcontract 9500008412.

References

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

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of Bleu in machine translation research. In *Proc. EACL 2006*, pages 249–256.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2007. (Meta-) evaluation of machine translation. In *Proc. Second Workshop on Statistical Machine Translation*, pages 136–158.

Yee Seng Chan and Hwee Tou Ng. 2008. MaxSim: A maximum similarity metric for machine translation evaluation. In *Proc. ACL-08: HLT*, pages 55–62.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proc. ACL 2005*, pages 531–540.

Markus Dreyer, Keith Hall, and Sanjeev Khudanpur. 2007. Comparing reordering constraints for SMT using efficient Bleu oracle computation. In *Proc. 2007 Workshop on Syntax and Structure in Statistical Translation*, pages 103–110.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proc. HLT-NAACL 2004*, pages 273–280.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proc. COLING-ACL 2006*, pages 961–968.

Jesús Giménez and Lluís Márquez. 2007. Linguistic features for automatic evaluation of heterogeneous MT systems. In *Proc. Second Workshop on Statistical Machine Translation*, pages 256–264.

Philipp Koehn. 2004a. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proc. AMTA 2004*, pages 115–124.

Philipp Koehn. 2004b. Statistical significance tests for machine translation evaluation. In *Proc. EMNLP 2004*, pages 388–395.

Shankar Kumar and William Byrne. 2004. Minimum Bayes-risk decoding for statistical machine translation. In *Proc. HLT-NAACL 2004*, pages 169–176.

Alon Lavie and Abhaya Agarwal. 2007. METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proc. Second Workshop on Statistical Machine Translation*, pages 228–231.

Gregor Leusch, Evgeny Matusov, and Hermann Ney. 2008. Complexity of finding the BLEU-optimal hypothesis in a confusion network. In *Proc. EMNLP 2008*. This volume.

Chi-Ho Li, Dongdong Zhang, Ming Zhou, and Hailei Zhang. 2008. An empirical study in source word deletion for phrase-based statistical machine translation. In *Proc. Third Workshop on Statistical Machine Translation*, pages 1–8.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proc. COLING-ACL 2006*, pages 761–768.

Chin-Yew Lin and Franz Josef Och. 2004. ORANGE: a method for evaluating automatic evaluation metrics for machine translation. In *Proc. COLING 2004*, pages 501–507.

Iaian McCowan, Darren Moore, John Dines, Daniel Gatica-Perez, Mike Flynn, Pierre Wellner, and Hervé Bourlard. 2005. On the use of information retrieval measures for speech recognition evaluation. Research Report 04-73, IDIAP Research Institute.

I. Dan Melamed, Ryan Green, and Joseph P. Turian. 2003. Precision and recall of machine translation. In *Proc. HLT-NAACL 2003*, pages 61–63. Companion volume.

Sonia Nießen, Franz Josef Och, Gregor Leusch, and Hermann Ney. 2000. An evaluation tool for machine translation: Fast evaluation for MT research. In *Proc. LREC 2000*, pages 39–45.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. ACL 2003*, pages 160–167.

sys	BLEU	BLEU-SBP	METEOR	GTM	MAXSIM
1	29.6^{++}	28.0	53.1^{++}	45.5^{++}	40.7^{++}
2	29.3^{++}	27.8	52.2^{--}	44.8^{--}	39.6^{--}

Table 2: The sign test yields inconsistent results with BLEU but not with other metrics. Significances are relative to other system.

	mixed-genre		genre-specific		
test set	BLEU	length	BLEU	length	ΔBLEU
nw	47.9	1.14	51.1	0.98	+3.2
web	16.3	0.87	16.8	0.95	+0.5
nw+web	31.5	0.97	30.4	0.96	−1.1

Table 3: When performing two genre-specific max-BLEU trainings instead of a single mixed-genre training, we expect that improvements in the newswire (nw) and web subsets should result in a similar improvement in the combined test set (nw+web), but this is not the case. Key: length = length ratio relative to effective reference length.

test set	ΔMETEOR	ΔGTM	ΔMAXSIM
nw	−2.2	−1.3	−2.8
web	+0.8	+0.7	+1.3
nw+web	−0.7	−0.6	−0.2

Table 4: Contradictory effects of genre-specific training were not observed with other metrics.

	mixed-genre	genre-specific	
test set	BLEU-SBP	BLEU-SBP	ΔBLEU-SBP
nw	49.6	49.9	+0.3
web	15.3	15.7	+0.4
nw+web	29.3	29.5	+0.2

Table 5: When performing two genre-specific max-BLEU-SBP trainings instead of a single mixed-genre training, we find as expected that improvements in the newswire (nw) and web subsets correlate with a similar improvement in the combined test set (nw+web).

	dev		test	
deletion	del%	BLEU	del%	BLEU
none	0	37.7	0	39.3
word	8.4	38.6^{++}	7.7	40.1^{++}
sentence	10.2	37.7	8.6	39.1

Table 6: Use of word-deletion rules can improve the BLEU score, and use of sentence-deletion rules shows no significant degradation, even though they are used heavily. Significances are relative to baseline (no deletion); all other differences are not statistically significant.

	test			
deletion	METEOR	GTM	MAXSIM	4-GRR
none	59.2	41.0	45.6	18.7
word	57.9	41.9	45.0	18.6
sentence	57.2	41.3	44.0	17.1

Table 7: Word and sentence deletion are punished by most of the other metrics. All systems used max-BLEU training. Significance testing was not performed.

max-BLEU training

deletion	dev BLEU-SBP	test BLEU-SBP
none	35.3	36.9
word	35.8^{+}	37.1
sentence	33.0^{--}	34.5^{--}

max-BLEU-SBP training

	dev		test	
deletion	del%	BLEU-SBP	del%	BLEU-SBP
none	0	35.8	0	37.1
word	5.3	36.3^{+}	5.0	37.3
sentence	0.02	35.9	0	37.5

Table 8: BLEU-SBP severely punishes the max-BLEU-trained sentence-deletion system; when we perform max-BLEU-SBP training, word deletion occurs less frequently and sentence deletion is nearly unused. Significances are relative to baseline (no deletion); other differences are not statistically significant.

Key:		
	+, ++	significant improvement ($p < 0.05$ or $p < 0.01$, respectively)
	−, −−	significant degradation ($p < 0.05$ or $p < 0.01$, respectively)
	Δ*metric*	change in *metric* due to genre-specific training
	del%	percentage of words deleted

Franz Josef Och. 2008. The Google statistical machine translation system for the 2008 NIST MT Evaluation. Presentation at the NIST Open Machine Translation 2008 Evaluation Workshop.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. ACL 2002*, pages 311–318.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proc. AMTA 2006*, pages 223–231.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2004. Max-margin markov networks. In *Proc. NIPS 2003*.

Christoph Tillmann and Tong Zhang. 2006. A discriminative global training algorithm for statistical MT. In *Proc. COLING-ACL 2006*, pages 721–728.

Roy W. Tromble, Shankar Kumar, Franz Och, and Wolfgang Macherey. 2008. Lattice minimum Bayes-risk decoding for statistical machine translation. In *Proc. EMNLP 2008*. This volume.

Robert A. Wagner. 1974. Order-*n* correction for regular languages. *Communications of the ACM*, 17(5):265.

Taro Watanabe, Jun Suzuki, Hajime Tsukuda, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proc. EMNLP 2007*, pages 764–773.

Ying Zhang, Stephan Vogel, and Alex Waibel. 2004. Interpreting BLEU/NIST scores: How much improvement do we need to have a better system? In *Proc. LREC 2004*, pages 2051–2054.

Liang Zhou, Chin-Yew Lin, and Eduard Hovy. 2006. Reevaluating machine translation results with paraphase support. In *Proc. EMNLP 2006*, pages 77–84.

Lattice Minimum Bayes-Risk Decoding for Statistical Machine Translation

Roy W. Tromble[1] and **Shankar Kumar**[2] and **Franz Och**[2] and **Wolfgang Macherey**[2]

[1]Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218, USA
royt@jhu.edu

[2] Google Inc.
1600 Amphitheatre Pkwy.
Mountain View, CA 94043, USA
{shankarkumar,och,wmach}@google.com

Abstract

We present Minimum Bayes-Risk (MBR) decoding over translation lattices that compactly encode a huge number of translation hypotheses. We describe conditions on the loss function that will enable efficient implementation of MBR decoders on lattices. We introduce an approximation to the BLEU score (Papineni et al., 2001) that satisfies these conditions. The MBR decoding under this approximate BLEU is realized using Weighted Finite State Automata. Our experiments show that the Lattice MBR decoder yields moderate, consistent gains in translation performance over N-best MBR decoding on Arabic-to-English, Chinese-to-English and English-to-Chinese translation tasks. We conduct a range of experiments to understand why Lattice MBR improves upon N-best MBR and study the impact of various parameters on MBR performance.

1 Introduction

Statistical language processing systems for speech recognition, machine translation or parsing typically employ the Maximum A Posteriori (MAP) decision rule which optimizes the 0-1 loss function. In contrast, these systems are evaluated using metrics based on string-edit distance (Word Error Rate), n-gram overlap (BLEU score (Papineni et al., 2001)), or precision/recall relative to human annotations. Minimum Bayes-Risk (MBR) decoding (Bickel and Doksum, 1977) aims to address this mismatch by selecting the hypothesis that minimizes the expected error in classification. Thus it directly incorporates the loss function into the decision criterion. The approach has been shown to give improvements over the MAP classifier in many areas of natural language processing including automatic speech recognition (Goel and Byrne, 2000), machine translation (Kumar and Byrne, 2004; Zhang and Gildea, 2008), bilingual word alignment (Kumar and Byrne, 2002), and parsing (Goodman, 1996; Titov and Henderson, 2006; Smith and Smith, 2007).

In statistical machine translation, MBR decoding is generally implemented by re-ranking an N-best list of translations produced by a first-pass decoder; this list typically contains between 100 and $10,000$ hypotheses. Kumar and Byrne (2004) show that MBR decoding gives optimal performance when the loss function is matched to the evaluation criterion; in particular, MBR under the sentence-level BLEU loss function (Papineni et al., 2001) gives gains on BLEU. This is despite the fact that the sentence-level BLEU loss function is an approximation to the exact corpus-level BLEU.

A different MBR inspired decoding approach is pursued in Zhang and Gildea (2008) for machine translation using Synchronous Context Free Grammars. A forest generated by an initial decoding pass is rescored using dynamic programming to maximize the expected count of synchronous constituents in the tree that corresponds to the translation. Since each constituent adds a new 4-gram to the existing translation, this approach approximately maximizes the expected BLEU.

In this paper we explore a different strategy to perform MBR decoding over *Translation Lattices* (Ueffing et al., 2002) that compactly encode a huge number of translation alternatives relative to an N-best list. This is a model-independent approach in that the lattices could be produced by any statistical MT system — both phrase-based and syntax-based systems would work in this framework. We will introduce conditions on the loss functions that can be incorporated in Lattice MBR decoding. We describe an approximation to the BLEU score (Papineni et al., 2001) that will satisfy these conditions. Our Lattice MBR decoding is realized using Weighted Finite State Automata.

We expect Lattice MBR decoding to improve upon N-best MBR primarily because lattices contain many more candidate translations than the N-best list. This has been demonstrated in speech recognition (Goel and Byrne, 2000). We conduct a range of translation experiments to analyze lattice MBR and compare it with N-best MBR. An important aspect of our lattice MBR is the linear approximation to the BLEU score. We will show that MBR decoding under this score achieves a performance that is at least as good as the performance obtained under sentence-level BLEU score.

The rest of the paper is organized as follows. We review MBR decoding in Section 2 and give the formulation in terms of a gain function. In Section 3, we describe the conditions on the gain function for efficient decoding over a lattice. The implementation of lattice MBR with Weighted Finite State Automata is presented in Section 4. In Section 5, we introduce the corpus BLEU approximation that makes it possible to perform efficient lattice MBR decoding. An example of lattice MBR with a toy lattice is presented in Section 6. We present lattice MBR experiments in Section 7. A final discussion is presented in Section 8.

2 Minimum Bayes Risk Decoding

Minimum Bayes-Risk (MBR) decoding aims to find the candidate hypothesis that has the least expected loss under the probability model (Bickel and Doksum, 1977). We begin with a review of MBR decoding for Statistical Machine Translation (SMT).

Statistical MT (Brown et al., 1990; Och and Ney, 2004) can be described as a mapping of a word sequence F in the source language to a word sequence E in the target language; this mapping is produced by the MT decoder $\delta(F)$. If the reference translation E is known, the decoder performance can be measured by the loss function $L(E, \delta(F))$. Given such a loss function $L(E, E')$ between an automatic translation E' and the reference E, and an underlying probability model $P(E|F)$, the MBR decoder has the following form (Goel and Byrne, 2000; Kumar and Byrne, 2004):

$$\begin{aligned}\hat{E} &= \operatorname*{argmin}_{E' \in \mathcal{E}} R(E') \\ &= \operatorname*{argmin}_{E' \in \mathcal{E}} \sum_{E \in \mathcal{E}} L(E, E') P(E|F),\end{aligned}$$

where $R(E')$ denotes the Bayes risk of candidate translation E' under the loss function L.

If the loss function between any two hypotheses can be bounded: $L(E, E') \leq L_{max}$, the MBR decoder can be rewritten in terms of a gain function $G(E, E') = L_{max} - L(E, E')$:

$$\hat{E} = \operatorname*{argmax}_{E' \in \mathcal{E}} \sum_{E \in \mathcal{E}} G(E, E') P(E|F). \qquad (1)$$

We are interested in performing MBR decoding under a sentence-level BLEU score (Papineni et al., 2001) which behaves like a gain function: it varies between 0 and 1, and a larger value reflects a higher similarity. We will therefore use Equation 1 as the MBR decoder.

We note that $\mathcal{E}$ represents the space of translations. For N-best MBR, this space $\mathcal{E}$ is the N-best list produced by a baseline decoder. We will investigate the use of a translation lattice for MBR decoding; in this case, $\mathcal{E}$ will represent the set of candidates encoded in the lattice.

In general, MBR decoding can use different spaces for hypothesis selection and risk computation: argmax and the sum in Equation 1 (Goel, 2001). As an example, the hypothesis could be selected from the N-best list while the risk is computed based on the entire lattice. Therefore, the MBR decoder can be more generally written as follows:

$$\hat{E} = \operatorname*{argmax}_{E' \in \mathcal{E}_h} \sum_{E \in \mathcal{E}_e} G(E, E') P(E|F), \qquad (2)$$

where $\mathcal{E}_h$ refers to the *Hypothesis space* from where the translations are chosen, and $\mathcal{E}_e$ refers to the *Evidence space* that is used for computing the Bayes-risk. We will present experiments (Section 7) to show the relative importance of these two spaces.

3 Lattice MBR Decoding

We now present MBR decoding on translation lattices. A translation word lattice is a compact representation for very large N-best lists of translation hypotheses and their likelihoods. Formally, it is an acyclic Weighted Finite State Acceptor (WFSA) (Mohri, 2002) consisting of states and arcs representing transitions between states. Each arc is labeled with a word and a weight. Each path in the lattice, consisting of consecutive transitions beginning at the distinguished initial state and ending at a final state, expresses a candidate translation. Aggregation of the weights along the path[1] produces the weight of the path's candidate $H(E, F)$ according to the model. In our setting, this weight will imply the posterior probability of the translation E given the source sentence F:

$$P(E|F) = \frac{\exp(\alpha H(E, F))}{\sum_{E' \in \mathcal{E}} \exp(\alpha H(E', F))}. \tag{3}$$

The scaling factor $\alpha \in [0, \infty)$ flattens the distribution when $\alpha < 1$, and sharpens it when $\alpha > 1$.

Because a lattice may represent a number of candidates exponential in the size of its state set, it is often impractical to compute the MBR decoder (Equation 1) directly. However, if we can express the gain function G as a sum of *local* gain functions g_i, then we now show that Equation 1 can be refactored and the MBR decoder can be computed efficiently. We loosely call a gain function local if it can be applied to all paths in the lattice via WFSA intersection (Mohri, 2002) without significantly multiplying the number of states.

In this paper, we are primarily concerned with local gain functions that weight n-grams. Let $\mathcal{N} = \{w_1, \dots, w_{|\mathcal{N}|}\}$ be the set of n-grams and let a local gain function $g_w : \mathcal{E} \times \mathcal{E} \to \mathbb{R}$, for $w \in \mathcal{N}$, be as follows:

$$g_w(E, E') = \theta_w \#_w(E') \delta_w(E), \tag{4}$$

where θ_w is a constant, $\#_w(E')$ is the number of times that w occurs in E', and $\delta_w(E)$ is 1 if $w \in E$ and 0 otherwise. That is, g_w is θ_w times the number of occurrences of w in E', or zero if w does not occur in E. We first assume that the overall gain function $G(E, E')$ can then be written as a sum of local gain functions and a constant θ_0 times the length of the hypothesis E'.

$$\begin{aligned} G(E, E') &= \theta_0|E'| + \sum_{w \in \mathcal{N}} g_w(E, E') \qquad (5) \\ &= \theta_0|E'| + \sum_{w \in \mathcal{N}} \theta_w \#_w(E') \delta_w(E) \end{aligned}$$

Given a gain function of this form, we can rewrite the risk (sum in Equation 1) as follows

$$\begin{aligned} &\sum_{E \in \mathcal{E}} G(E, E') P(E|F) \\ = &\sum_{E \in \mathcal{E}} \Big(\theta_0|E'| + \sum_{w \in \mathcal{N}} \theta_w \#_w(E') \delta_w(E)\Big) P(E|F) \\ = &\ \theta_0|E'| + \sum_{w \in \mathcal{N}} \theta_w \#_w(E') \sum_{E \in \mathcal{E}_w} P(E|F), \end{aligned}$$

where $\mathcal{E}_w = \{E \in \mathcal{E} | \delta_w(E) > 0\}$ represents the paths of the lattice containing the n-gram w at least once. The MBR decoder on lattices (Equation 1) can therefore be written as

$$\hat{E} = \operatorname*{argmax}_{E' \in \mathcal{E}} \Big\{\theta_0|E'| + \sum_{w \in \mathcal{N}} \theta_w \#_w(E') p(w|\mathcal{E})\Big\}. \tag{6}$$

Here $p(w|\mathcal{E}) = \sum_{E \in \mathcal{E}_w} P(E|F)$ is the posterior probability of the n-gram w in the lattice. We have thus replaced a summation over a possibly exponential number of items ($E \in \mathcal{E}$) with a summation over the number of n-grams that occur in $\mathcal{E}$, which is at worst polynomial in the number of edges in the lattice that defines $\mathcal{E}$. We compute the posterior probability of each n-gram w as:

$$p(w|\mathcal{E}) = \sum_{E \in \mathcal{E}_w} P(E|F) = \frac{Z(\mathcal{E}_w)}{Z(\mathcal{E})}, \tag{7}$$

where $Z(\mathcal{E}) = \sum_{E' \in \mathcal{E}} \exp(\alpha H(E', F))$ (denominator in Equation 3) and $Z(\mathcal{E}_w) = \sum_{E' \in \mathcal{E}_w} \exp(\alpha H(E', F))$. $Z(\mathcal{E})$ and $Z(\mathcal{E}_w)$ represent the sums[2] of weights of all paths in the lattices $\mathcal{E}_w$ and $\mathcal{E}$ respectively.

4 WFSA MBR Computations

We now show how the Lattice MBR Decision Rule (Equation 6) can be implemented using Weighted Finite State Automata (Mohri, 1997). There are four steps involved in decoding starting from weighted finite-state automata representing the candidate outputs of a translation system. We will describe these

[1] using the log semiring's *extend* operator

[2] in the log semiring, where $\log +(x, y) = \log(e^x + e^y)$ is the *collect* operator (Mohri, 2002)

steps in the setting where the evidence lattice $\mathcal{E}_e$ may be different from the hypothesis lattice $\mathcal{E}_h$ (Equation 2).

1. Extract the set of n-grams that occur in the evidence lattice $\mathcal{E}_e$. For the usual BLEU score, n ranges from one to four.

2. Compute the posterior probability $p(w|\mathcal{E})$ of each of these n-grams.

3. Intersect each n-gram w, with an appropriate weight (from Equation 6), to an initially unweighted copy of the hypothesis lattice $\mathcal{E}_h$.

4. Find the best path in the resulting automaton.

Computing the set of n-grams $\mathcal{N}$ that occur in a finite automaton requires a traversal, in topological order, of all the arcs in the automaton. Because the lattice is acyclic, this is possible. Each state q in the automaton has a corresponding set of n-grams $\mathcal{N}_q$ ending there.

1. For each state q, $\mathcal{N}_q$ is initialized to $\{\epsilon\}$, the set containing the empty n-gram.

2. Each arc in the automaton extends each of its source state's n-grams by its word label, and adds the resulting n-grams to the set of its target state. (ϵ arcs do not extend n-grams, but transfer them unchanged.) n-grams longer than the desired order are discarded.

3. $\mathcal{N}$ is the union over all states q of $\mathcal{N}_q$.

Given an n-gram, w, we construct an automaton matching any path containing the n-gram, and intersect that automaton with the lattice to find the set of paths containing the n-gram ($\mathcal{E}_w$ in Equation 7). Suppose $\mathcal{E}$ represent the weighted lattice, we compute[3]: $\mathcal{E}_w = \mathcal{E} \cap (\overline{w}\ w\ \Sigma^*)$, where $\overline{w} = (\overline{\Sigma^*\ w\ \Sigma^*})$ is the language that contains all strings that do not contain the n-gram w. The posterior probability $p(w|\mathcal{E})$ of n-gram w can be computed as a ratio of the total weights of paths in $\mathcal{E}_w$ to the total weights of paths in the original lattice (Equation 7).

For each n-gram $w \in \mathcal{N}$, we then construct an automaton that accepts an input E with weight equal to the product of the number of times the n-gram occurs in the input ($\#_w(E)$), the n-gram factor θ_w from Equation 6, and the posterior probability $p(w|\mathcal{E})$. The automaton corresponds to the weighted regular expression (Karttunen et al., 1996): $\bar{w}(w/(\theta_w p(w|\mathcal{E}))\ \bar{w})^*$.

We successively intersect each of these automata with an automaton that begins as an unweighted copy of the lattice $\mathcal{E}_h$. This automaton must also incorporate the factor θ_0 of each word. This can be accomplished by intersecting the unweighted lattice with the automaton accepting $(\Sigma/\theta_0)^*$. The resulting MBR automaton computes the total expected gain of each path. A path in this automaton that corresponds to the word sequence E' has cost: $\theta_0|E'| + \sum_{w\in\mathcal{N}} \theta_w \#_w(E) p(w|\mathcal{E})$ (expression within the curly brackets in Equation 6).

Finally, we extract the best path from the resulting automaton[4], giving the lattice MBR candidate translation according to the gain function (Equation 6).

5 Linear Corpus BLEU

Our Lattice MBR formulation relies on the decomposition of the overall gain function as a sum of local gain functions (Equation 5). We here describe a linear approximation to the log(BLEU score) (Papineni et al., 2001) which allows such a decomposition. This will enable us to rewrite the log(BLEU) as a linear function of n-gram matches and the hypothesis length. Our strategy will be to use a first order Taylor-series approximation to what we call the corpus log(BLEU) gain: the change in corpus log(BLEU) contributed by the sentence relative to not including that sentence in the corpus.

Let r be the reference length of the corpus, c_0 the candidate length, and $\{c_n|1 \leq n \leq 4\}$ the number of n-gram matches. Then, the corpus BLEU score $B(r, c_0, c_n)$ can be defined as follows (Papineni et al., 2001):

$$\begin{aligned} \log B &= \min\left(0, 1 - \frac{r}{c_0}\right) + \frac{1}{4}\sum_{n=1}^{4} \log \frac{c_n}{c_0 - \Delta_n}, \\ &\approx \min\left(0, 1 - \frac{r}{c_0}\right) + \frac{1}{4}\sum_{n=1}^{4} \log \frac{c_n}{c_0}, \end{aligned}$$

where we have ignored Δ_n, the difference between the number of words in the candidate and the num-

[3]in the log semiring (Mohri, 2002)

[4]in the $(\max, +)$ semiring (Mohri, 2002)

ber of n-grams. If L is the average sentence length in the corpus, $\Delta_n \approx (n-1)\frac{c_0}{L}$.

The corpus log(BLEU) gain is defined as the change in log(BLEU) when a new sentence's (E') statistics are added to the corpus statistics:

$$G = \log B' - \log B,$$

where the counts in B' are those of B plus those for the current sentence. We will assume that the brevity penalty (first term in the above approximation) does not change when adding the new sentence. In experiments not reported here, we found that taking into account the brevity penalty at the sentence level can cause large fluctuations in lattice MBR performance on different test sets. We therefore treat only c_ns as variables.

The corpus log BLEU gain is approximated by a first-order vector Taylor series expansion about the initial values of c_n.

$$G \approx \sum_{n=0}^{N} (c'_n - c_n) \left. \frac{\partial \log B'}{\partial c'_n} \right|_{c'_n = c_n}, \qquad (8)$$

where the partial derivatives are given by

$$\begin{aligned} \frac{\partial \log B}{\partial c_0} &= \frac{-1}{c_0}, \\ \frac{\partial \log B}{\partial c_n} &= \frac{1}{4c_n}. \end{aligned} \qquad (9)$$

Substituting the derivatives in Equation 8 gives

$$G = \Delta \log B \approx -\frac{\Delta c_0}{c_0} + \frac{1}{4}\sum_{n=1}^{4} \frac{\Delta c_n}{c_n}, \qquad (10)$$

where each $\Delta c_n = c'_n - c_n$ counts the statistic in the sentence of interest, rather than the corpus as a whole. This score is therefore a linear function in counts of words Δc_0 and n-gram matches Δc_n. Our approach ignores the count clipping present in the exact BLEU score where a correct n-gram present once in the reference but several times in the hypothesis will be counted only once as correct. Such an approach is also followed in Dreyer et al. (2007).

Using the above first-order approximation to gain in log corpus BLEU, Equation 9 implies that θ_0, θ_w from Section 3 would have the following values:

$$\begin{aligned} \theta_0 &= \frac{-1}{c_0} \\ \theta_w &= \frac{1}{4c_{|w|}}. \end{aligned} \qquad (11)$$

5.1 N-gram Factors

We now describe how the n-gram factors (Equation 11) are computed. The factors depend on a set of n-gram matches and counts (c_n; $n \in \{0, 1, 2, 3, 4\}$). These factors could be obtained from a decoding run on a development set. However, doing so could make the performance of lattice MBR very sensitive to the actual BLEU scores on a particular run. We would like to avoid such a dependence and instead, obtain a set of parameters which can be estimated from multiple decoding runs without MBR. To achieve this, we make use of the properties of n-gram matches. It is known that the average n-gram precisions decay approximately exponentially with n (Papineni et al., 2001). We now assume that the number of matches of each n-gram is a constant ratio r times the matches of the corresponding $n-1$ gram.

If the unigram precision is p, we can obtain the n-gram factors ($n \in \{1, 2, 3, 4\}$) (Equation 11) as a function of the parameters p and r, and the number of unigram tokens T:

$$\begin{aligned} \theta_0 &= \frac{-1}{T} \\ \theta_n &= \frac{1}{4Tp \times r^{n-1}} \end{aligned} \qquad (12)$$

We set p and r to the average values of unigram precision and precision ratio across multiple development sets. Substituting the above factors in Equation 6, we find that the MBR decision does not depend on T; therefore any value of T can be used.

6 An Example

Figure 1 shows a toy lattice and the final MBR automaton (Section 4) for BLEU with a maximum n-gram order of 2. We note that the MBR hypothesis (*bcde*) has a higher decoder cost relative to the MAP hypothesis (*abde*). However, *bcde* gets a higher expected gain (Equation 6) than *abde* since it shares more n-grams with the Rank-3 hypothesis (*bcda*). This illustrates how a lattice can help select MBR translations that can differ from the MAP translation.

7 Experiments

We now present experiments to evaluate MBR decoding on lattices under the linear corpus BLEU

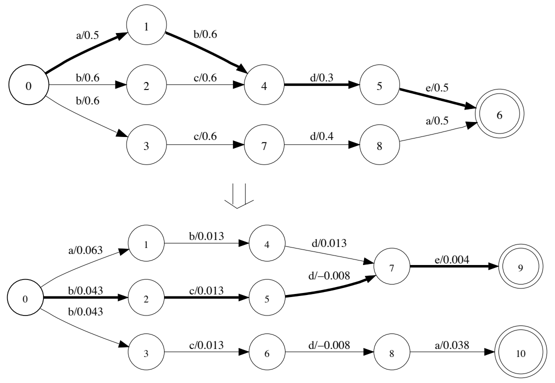


Figure 1: An example translation lattice with decoder costs (top) and its MBR Automaton for BLEU-2 (bottom). The bold path in the top is the MAP hypothesis and the bold path in the bottom is the MBR hypothesis. The precision parameters in Equation 12 are set to: $T = 10, p = 0.85, r = 0.72$.

Dataset	# of sentences		
	aren	zhen	enzh
dev1	1353	1788	1664
dev2	663	919	919
blind	1360	1357	1859

Table 1: Statistics over the development and test sets.

gain. We start with a description of the data sets and the SMT system.

7.1 Development and Blind Test Sets

We present our experiments on the constrained data track of the NIST 2008 Arabic-to-English (aren), Chinese-to-English (zhen), and English-to-Chinese (enzh) machine translation tasks.[5] In all language pairs, the parallel and monolingual data consists of all the allowed training sets in the constrained track.

For each language pair, we use two development sets: one for Minimum Error Rate Training (Och, 2003; Macherey et al., 2008), and the other for tuning the scale factor for MBR decoding. Our development sets consists of the NIST 2004/2003 evaluation sets for both aren and zhen, and NIST 2006 (NIST portion)/2003 evaluation sets for enzh. We report results on NIST 2008 which is our blind test set. Statistics computed over these data sets are reported in Table 1.

[5]http://www.nist.gov/speech/tests/mt/

7.2 MT System Description

Our phrase-based statistical MT system is similar to the alignment template system described in Och and Ney (2004). The system is trained on parallel corpora allowed in the constrained track. We first perform sentence and sub-sentence chunk alignment on the parallel documents. We then train word alignment models (Och and Ney, 2003) using 6 Model-1 iterations and 6 HMM iterations. An additional 2 iterations of Model-4 are performed for zhen and enzh pairs. Word Alignments in both source-to-target and target-to-source directions are obtained using the Maximum A-Posteriori (MAP) framework (Matusov et al., 2004). An inventory of phrase-pairs up to length 5 is then extracted from the union of source-target and target-source alignments. Several feature functions are then computed over the phrase-pairs. 5-gram word language models are trained on the allowed monolingual corpora. Minimum Error Rate Training under BLEU is used for estimating approximately 20 feature function weights over the dev1 development set.

Translation is performed using a standard dynamic programming beam-search decoder (Och and Ney, 2004) using two decoding passes. The first decoder pass generates either a lattice or an N-best list. MBR decoding is performed in the second pass. The MBR scaling parameter (α in Equation 3) is tuned on the dev2 development set.

7.3 Translation Results

We next report translation results from lattice MBR decoding. All results will be presented on the NIST 2008 evaluation sets. We report results using the NIST implementation of the BLEU score which computes the brevity penalty using the shortest reference translation for each segment (NIST, 2002 2008). The BLEU scores are reported at the word-level for aren and zhen but at the character level for enzh. We measure statistical significance using 95% confidence intervals computed with paired bootstrap resampling (Koehn, 2004). In all tables, systems in a column show statistically significant differences unless marked with an asterisk.

We first compare lattice MBR to N-best MBR decoding and MAP decoding (Table 2). In these experiments, we hold the likelihood scaling factor α a

	BLEU(%)		
	aren	zhen	enzh
MAP	43.7	27.9	41.4
N-best MBR	43.9	28.3^*	42.0
Lattice MBR	44.9	28.5^*	42.6

Table 2: Lattice MBR, N-best MBR & MAP decoding. On zhen, Lattice MBR and N-best MBR do not show statistically significant differences.

	BLEU(%)		
	aren	zhen	enzh
Lattice MBR, Lin. Corpus BLEU	44.2	28.1	42.2
N-best MBR, Sent. BLEU	43.9^*	28.3^*	42.0^*
N-best MBR, Sent. Log BLEU	44.0^*	28.3^*	41.9^*

Table 3: Lattice and N-best MBR (with Sentence BLEU/Sentence log BLEU) on a 1000-best list. In each column, entries with an asterisk do not show statistically significant differences.

		BLEU(%)		
Hyp Space	Evid Space	aren	zhen	enzh
Lattice	Lattice	44.9	28.5	42.6
1000-best	Lattice	44.6	28.5	42.6
Lattice	1000-best	44.1^*	28.0^*	42.1
1000-best	1000-best	44.2^*	28.1^*	42.2

Table 4: Lattice MBR with restrictions on hypothesis and evidence spaces. In each column, entries with an asterisk do not show statistically significant differences.

constant; it is set to 0.2 for aren and enzh, and 0.1 for zhen. The translation lattices are pruned using Forward-Backward pruning (Sixtus and Ortmanns, 1999) so that the average numbers of arcs per word (lattice density) is 30. For N-best MBR, we use N-best lists of size 1000. To match the loss function, Lattice MBR is performed at the word level for aren/zhen and at the character level for enzh. Our lattice MBR is implemented using the Google OpenFst library.[6] In our experiments, p, r (Equation 12) have values of 0.85/0.72, 0.80/0.62, and 0.63/0.48 for aren, zhen, and enzh respectively.

We note that Lattice MBR provides gains of 0.2-1.0 BLEU points over N-best MBR, which in turn gives 0.2-0.6 BLEU points over MAP. These gains are obtained on top of a baseline system that has competitive performance relative to the results reported in the NIST 2008 Evaluation.[7] This demonstrates the effectiveness of lattice MBR decoding as a realization of MBR decoding which yields substantial gains over the N-best implementation.

The gains from lattice MBR over N-best MBR could be due to a combination of factors. These include: 1) better approximation of the corpus BLEU score, 2) larger hypothesis space, and 3) larger evidence space. We now present experiments to tease apart these factors.

Our first experiment restricts both the hypothesis and evidence spaces in lattice MBR to the 1000-best list (Table 3). We compare this to N-best MBR with: a) sentence-level BLEU, and b) sentence-level log BLEU.

The results show that when restricted to the 1000-best list, Lattice MBR performs slightly better than N-best MBR (with sentence BLEU) on aren/enzh while N-best MBR is better on zhen. We hypothesize that on aren/enzh, the linear corpus BLEU gain (Equation 10) is better correlated to the actual corpus BLEU than sentence-level BLEU while the opposite is true on zhen. N-best MBR gives similar results with either sentence BLEU or sentence log BLEU. This confirms that using a log BLEU score does not change the outcome of MBR decoding and further justifies our Taylor-series approximation of the log BLEU score.

We next attempt to understand factors 2 and 3. To do that, we carry out lattice MBR when either the hypothesis or the evidence space in Equation 2 is restricted to 1000-best hypotheses (Table 4). For comparison, we also include results from lattice MBR when both hypothesis and evidence spaces are identical: either the full lattice or the 1000-best list (from Tables 2 and 3).

These results show that lattice MBR results are almost unchanged when the hypothesis space is restricted to a 1000-best list. However, when the evidence space is shrunk to a 1000-best list, there is a significant degradation in performance; these latter results are almost identical to the scenario when both evidence and hypothesis spaces are restricted to the 1000-best list. This experiment throws light on what makes lattice MBR effective over N-best MBR. Relative to the N-best list, the translation lattice provides a better estimate of the expected BLEU score. On the other hand, there are few hypotheses

[6]http://www.openfst.org/

[7] http://www.nist.gov/speech/tests/mt/2008/doc/mt08_official_results_v0.html

outside the 1000-best list which are selected by lattice MBR.

Finally, we show how the performance of lattice MBR changes as a function of the lattice density. The lattice density is the average number of arcs per word and can be varied using Forward-Backward pruning (Sixtus and Ortmanns, 1999). Figure 2 reports the average number of lattice paths and BLEU scores as a function of lattice density. The results show that Lattice MBR performance generally improves when the size of the lattice is increased. However, on zhen, there is a small drop beyond a density of 10. This could be due to low quality (low posterior probability) hypotheses that get included at the larger densities and result in a poorer estimate of the expected BLEU score. On aren and enzh, there are some gains beyond a lattice density of 30. These gains are relatively small and come at the expense of higher memory usage; we therefore work with a lattice density of 30 in all our experiments. We note that Lattice MBR is operating over lattices which are gigantic in comparison to the number of paths in an N-best list. At a lattice density of 30, the lattices in aren contain on an average about 10^{81} hypotheses!

7.4 Lattice MBR Scale Factor

We next examine the role of the scale factor α in lattice MBR decoding. The MBR scale factor determines the flatness of the posterior distribution (Equation 3). It is chosen using a grid search on the dev2 set (Table 1). Figure 3 shows the variation in BLEU scores on eval08 as this parameter is varied. The results show that it is important to tune this factor. The optimal scale factor is identical for all three language pairs. In experiments not reported in this paper, we have found that the optimal scaling factor on a moderately sized development set carries over to unseen test sets.

7.5 Maximum n-gram Order

Lattice MBR Decoding (Equation 6) involves computing a posterior probability for each n-gram in the lattice. We would like to speed up the Lattice MBR computation (Section 4) by restricting the maximum order of the n-grams in the procedure. The results (Table 5) show that on aren, there is no degradation if we limit the maximum order of the n-grams to 3. However, on zhen/enzh, there is improvement by considering 4-grams. We can therefore reduce Lattice MBR computations in aren.

	BLEU(%)		
Max n-gram order	aren	zhen	enzh
1	38.7	26.8	40.0
2	44.1	27.4	42.2
3	44.9	28.0	42.4
4	44.9	28.5	42.6

Table 5: Lattice MBR as a function of max n-gram order.

8 Discussion

We have presented a procedure for performing Minimum Bayes-Risk Decoding on translation lattices. This is a significant development in that the MBR decoder operates over a very large number of translations. In contrast, the current N-best implementation of MBR can be scaled to, at most, a few thousands of hypotheses. If the number of hypotheses is greater than, say 20,000, the N-best MBR becomes computationally expensive. The lattice MBR technique is efficient when performed over enormous number of hypotheses (up to 10^{80}) since it takes advantage of the compact structure of the lattice. Lattice MBR gives consistent improvements in translation performance over N-best MBR decoding, which is used in many state-of-the-art research translation systems. Moreover, we see gains on three different language pairs.

There are two potential reasons why Lattice MBR decoding could outperform N-best MBR: a larger hypothesis space from which translations could be selected or a larger evidence space for computing the expected loss. Our experiments show that the main improvement comes from the larger evidence space: a larger set of translations in the lattice provides a better estimate of the expected BLEU score. In other words, the lattice provides a better posterior distribution over translation hypotheses relative to an N-best list. This is a novel insight into the workings of MBR decoding. We believe this could be possibly employed when designing discriminative training approaches for machine translation. More generally, we have found a component in machine translation where the posterior distribution over hypotheses plays a crucial role.

We have shown the effect of the MBR scaling fac-

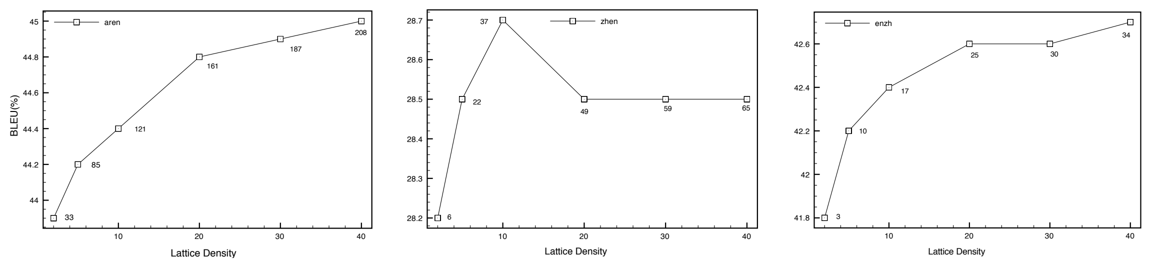


Figure 2: Lattice MBR vs. lattice density: aren/zhen/enzh. Each point also shows the $\log_e$(Avg. # of paths).

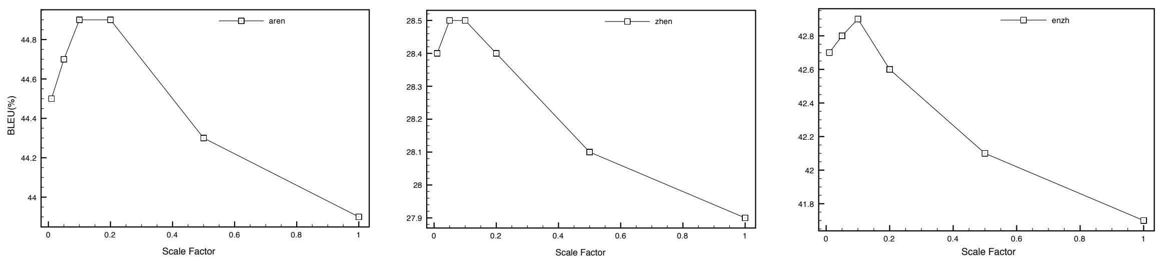


Figure 3: Lattice MBR with various scale factors α: aren/zhen/enzh.

tor on the performance of lattice MBR. The scale factor determines the flatness of the posterior distribution over translation hypotheses. A scale of 0.0 means a uniform distribution while 1.0 implies that there is no scaling. This is an important parameter that needs to be tuned on a development set. There has been prior work in MBR speech recognition and machine translation (Goel and Byrne, 2000; Ehling et al., 2007) which has shown the need for tuning this factor. Our MT system parameters are trained with Minimum Error Rate Training which assigns a very high posterior probability to the MAP translation. As a result, it is necessary to flatten the probability distribution so that MBR decoding can select hypotheses other than the MAP hypothesis.

Our Lattice MBR implementation is made possible due to the linear approximation of the BLEU score. This linearization technique has been applied elsewhere when working with BLEU: Smith and Eisner (2006) approximate the expectation of log BLEU score. In both cases, a linear metric makes it easier to compute the expectation. While we have applied lattice MBR decoding to the approximate BLEU score, we note that our procedure (Section 3) is applicable to other gain functions which can be decomposed as a sum of local gain functions. In particular, our framework might be useful with translation metrics such as TER (Snover et al., 2006) or METEOR (Lavie and Agarwal, 2007).

In contrast to a phrase-based SMT system, a syntax based SMT system (e.g. Zollmann and Venugopal (2006)) can generate a hypergraph that represents a generalized translation lattice with words and hidden tree structures. We believe that our lattice MBR framework can be extended to such hypergraphs with loss functions that take into account both BLEU scores as well as parse tree structures.

Lattice and Forest based search and training procedures are not yet common in statistical machine translation. However, they are promising because the search space of translations is much larger than the typical N-best list (Mi et al., 2008). We hope that our approach will provide some insight into the design of lattice-based search procedures along with the use of non-linear, global loss functions such as BLEU.

References

P. J. Bickel and K. A. Doksum. 1977. *Mathematical Statistics: Basic Ideas and Selected topics*. Holden-Day Inc., Oakland, CA, USA.

P. F. Brown, J. Cocke, S. A. Della Pietra, V. J . Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S.

Roossin. 1990. A Statistical Approach to Machine Translation. *Computational Linguistics*, 16(2):79–85.

M. Dreyer, K. Hall, and S. Khudanpur. 2007. Comparing Reordering Constraints for SMT Using Efficient BLEU Oracle Computation. In *SSST, NAACL-HLT 2007*, pages 103–110, Rochester, NY, USA, April.

N. Ehling, R. Zens, and H. Ney. 2007. Minimum Bayes Risk Decoding for BLEU. In *ACL 2007*, pages 101–104, Prague, Czech Republic, June.

V. Goel and W. Byrne. 2000. Minimum Bayes-Risk Automatic Speech Recognition. *Computer Speech and Language*, 14(2):115–135.

V. Goel. 2001. *Minimum Bayes-Risk Automatic Speech Recognition*. Ph.D. thesis, Johns Hopkins University, Baltimore, MD, USA.

J. Goodman. 1996. Parsing Algorithms and Metrics. In *ACL*, pages 177–183, Santa Cruz, CA, USA.

L. Karttunen, J-p. Chanod, G. Grefenstette, and A. Schiller. 1996. Regular Expressions for Language Engineering. *Natural Language Engineering*, 2:305–328.

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

S. Kumar and W. Byrne. 2002. Minimum Bayes-Risk word alignments of bilingual texts. In *EMNLP*, pages 140–147, Philadelphia, PA, USA.

S. Kumar and W. Byrne. 2004. Minimum Bayes-Risk Decoding for Statistical Machine Translation. In *HLT-NAACL*, pages 169–176, Boston, MA, USA.

A. Lavie and A. Agarwal. 2007. METEOR: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments. In *SMT Workshop, ACL*, pages 228–231, Prague, Czech Republic.

W. Macherey, F. Och, I. Thayer, and J. Uszkoreit. 2008. Lattice-based Minimum Error Rate Training for Statistical Machine Translation. In *EMNLP*, Honolulu, Hawaii, USA.

E. Matusov, R. Zens, and H. Ney. 2004. Symmetric Word Alignments for Statistical Machine Translation. In *COLING*, Geneva, Switzerland.

H. Mi, L. Huang, and Q. Liu. 2008. Forest-Based Translation. In *ACL*, Columbus, OH, USA.

M. Mohri. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(3).

M. Mohri. 2002. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350.

NIST. 2002-2008. The NIST Machine Translation Evaluations. http://www.nist.gov/speech/tests/mt/.

F. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19 – 51.

F. Och and H. Ney. 2004. The Alignment Template Approach to Statistical Machine Translation. *Computational Linguistics*, 30(4):417 – 449.

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

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

A. Sixtus and S. Ortmanns. 1999. High Quality Word Graphs Using Forward-Backward Pruning. In *ICASSP*, Phoenix, AZ, USA.

D. Smith and J. Eisner. 2006. Minimum Risk Annealing for Training Log-Linear Models. In *ACL*, Sydney, Australia.

D. Smith and N. Smith. 2007. Probabilistic models of nonprojective dependency trees. In *EMNLP-CoNLL*, Prague, Czech Republic.

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *AMTA*, Boston, MA, USA.

I. Titov and J. Henderson. 2006. Loss Minimization in Parse Reranking. In *EMNLP*, Sydney, Australia.

N. Ueffing, F. Och, and H. Ney. 2002. Generation of Word Graphs in Statistical Machine Translation. In *EMNLP*, Philadelphia, PA, USA.

H. Zhang and D. Gildea. 2008. Efficient Multi-pass Decoding for Synchronous Context Free Grammars. In *ACL*, Columbus, OH, USA.

A. Zollmann and A. Venugopal. 2006. Syntax Augmented Machine Translation via Chart Parsing. In *HLT-NAACL*, New York, NY, USA.

Phrase Translation Probabilities with ITG Priors and Smoothing as Learning Objective

Markos Mylonakis
Language and Computation, ILLC
Faculty of Science
University of Amsterdam
m.mylonakis@uva.nl

Khalil Sima'an
Language and Computation, ILLC
Faculty of Science
University of Amsterdam
k.simaan@uva.nl

Abstract

The conditional phrase translation probabilities constitute the principal components of phrase-based machine translation systems. These probabilities are estimated using a heuristic method that does not seem to optimize any reasonable objective function of the word-aligned, parallel training corpus. Earlier efforts on devising a better understood estimator either do not scale to reasonably sized training data, or lead to deteriorating performance. In this paper we explore a new approach based on three ingredients (1) A generative model with a prior over latent segmentations derived from Inversion Transduction Grammar (ITG), (2) A phrase table containing all phrase pairs without length limit, and (3) Smoothing as learning objective using a novel Maximum-A-Posteriori version of Deleted Estimation working with Expectation-Maximization. Where others conclude that latent segmentations lead to overfitting and deteriorating performance, we show here that these three ingredients give performance equivalent to the heuristic method *on reasonably sized training data*.

1 Motivation

A major component in phrase-based statistical Machine translation (PBSMT) (Zens et al., 2002; Koehn et al., 2003) is the table of conditional probabilities of phrase translation pairs. The pervading method for estimating these probabilities is a simple heuristic based on the relative frequency of the phrase pair *in the multi-set of the phrase pairs extracted from the word-aligned corpus* (Koehn et al., 2003). While this heuristic estimator gives good empirical results, it does not seem to optimize any intuitively reasonable objective function of the (word-aligned) parallel corpus (see e.g., (DeNero et al., 2006)) The mounting number of efforts attacking this problem over the last few years (DeNero et al., 2006; Marcu and Wong, 2002; Birch et al., 2006; Moore and Quirk, 2007; Zhang et al., 2008) exhibits its difficulty. So far, none has lead to an alternative method that performs as well as the heuristic on *reasonably sized data (approx. 1000k sentence pair)*.

Given a parallel corpus, an estimator for phrase-tables in PBSMT involves two interacting decisions (1) which phrase pairs to extract, and (2) how to assign probabilities to the extracted pairs. The heuristic estimator employs word-alignment (Giza++) (Och and Ney, 2003) and a few thumb rules for defining phrase pairs, and then extracts a multi-set of phrase pairs and estimates their conditional probabilities based on the counts in the multi-set. Using this method for extracting a set of phrase pairs, (DeNero et al., 2006; Moore and Quirk, 2007) aim at defining a better estimator for the probabilities. Generally speaking, both efforts report deteriorating translation performance relative to the heuristic.

Instead of employing word-alignment to guide phrase pair extraction, it is theoretically more appealing to aim at phrase alignment as part of the estimation process (Marcu and Wong, 2002; Birch et al., 2006). This way, phrase pair extraction goes hand-in-hand with estimating the probabilities. However, in practice, due to the huge number of possible phrase pairs, this task is rather challenging, both computationally and statistically. It is hard to define both a manageable phrase pair translation model and a well-founded training regime that would scale up to reasonably sized parallel corpora (see e.g., (Birch et al., 2006)). It remains to be seen whether this theoretically interesting approach will lead to improved phrase probability estimates.

In this paper we also start out from a standard phrase extraction procedure based on word-alignment and aim solely at estimating the conditional probabilities for the phrase pairs and their reverse translation probabilities. Unlike preceding work, we extract *all phrase pairs* from the training corpus and estimate their probabilities, i.e., without limit on length. We present a novel formulation of a conditional translation model that works with a *prior over segmentations* and a bag of conditional phrase pairs. We use binary Synchronous Context-Free Grammar (bSCFG), based on Inversion Transduction Grammar (ITG) (Wu, 1997; Chiang, 2005a), to define the set of eligible segmentations for an aligned sentence pair. We also show how the number of spurious derivations per segmentation in this bSCFG can be used for devising a prior probability over the space of segmentations, capturing the bias *in the data* towards monotone translation. The heart of the estimation process is a new *smoothing estimator*, a penalized version of Deleted Estimation, which averages the temporary **probability estimates** of multiple parallel EM processes at each joint iteration.

For evaluation we use a state-of-the-art baseline system (Moses) (Hoang and Koehn, 2008) which works with a log-linear interpolation of feature functions optimized by MERT (Och, 2003). We simply substitute our own estimates for the heuristic phrase translation estimates (both directions and the phrase penalty score) and compare the two within the Moses decoder. While our estimates differ substantially from the heuristic, their performance is on par with the heuristic estimates. This is remarkable given the fact that comparable previous work (DeNero et al., 2006; Moore and Quirk, 2007) did not match the performance of the heuristic estimator using large training sets. We find that smoothing is crucial for achieving good estimates. This is in line with earlier work on consistent estimation for similar models (Zollmann and Sima'an, 2006), and agrees with the most up-to-date work that employs Bayesian priors over the estimates (Zhang et al., 2008).

2 Related work

Marcu and Wong (Marcu and Wong, 2002) realize that the problem of extracting phrase pairs should be intertwined with the method of probability estimation. They formulate a joint phrase-based model in which a source-target sentence pair is generated jointly. However, the huge number of possible *phrase-alignments* prohibits scaling up the estimation by Expectation-Maximization (EM) (Dempster et al., 1977) to large corpora. Birch et al (Birch et al., 2006) provide soft measures for including word-alignments in the estimation process and obtain improved results only on small data sets.

Coming up-to-date, (Blunsom et al., 2008) attempt a related estimation problem to (Marcu and Wong, 2002), using the expanded phrase pair set of (Chiang, 2005a), working with an exponential model and concentrating on marginalizing out the latent segmentation variable. Also most up-to-date, (Zhang et al., 2008) report on a multi-stage model, *without* a latent segmentation variable, but with a strong prior preferring sparse estimates embedded in a Variational Bayes (VB) estimator and concentrating the efforts on pruning both the space of phrase pairs and the space of (ITG) analyses. The latter two efforts report improved performance, albeit again on a limited training set (approx. 140k sentences up to a certain length).

DeNero et al (2006) have explored estimation using EM of phrase pair probabilities under a conditional translation model based on the original source-channel formulation. This model involves a hidden segmentation variable that is set uniformly (or to prefer shorter phrases over longer ones). Furthermore, the model involves a reordering component akin to the one used in IBM model 3. Despite this, the heuristic estimator remains superior because ”EM learns overly determinized segmentations and translation parameters, overfitting the training data and failing to generalize”. More recently, (Moore and Quirk, 2007) devise a estimator working with a model that does not include a hidden segmentation variable but works with a heuristic iterative procedure (rather than MLE or EM). The translation results remain inferior to the heuristic but the authors note an interesting trade-off between decoding speed and the various settings of this estimator.

Our work expands on the general approach taken by (DeNero et al., 2006; Moore and Quirk, 2007) but arrives at insights similar to those of the most recent work (Zhang et al., 2006), albeit in a completely different manner. The present work differs from all preceding work in that it employs the set of *all phrase pairs* during training. It differs from (Zhang et al., 2008) in that it does postulate a latent segmentation variable and puts the prior directly over that variable rather than over the ITG synchronous rule estimates. Our method neither excludes phrase pairs before estimation nor does it prune the space of possible segmentations/analyses during training/estimation. As well as smoothing, we find (in the same vein as (Zhang et al., 2008)) that setting effective priors/smoothing is crucial for EM to arrive at better estimates.

3 The Translation Model

Given a word-aligned parallel corpus of source-target sentences, it is common practice to extract a set of phrase pairs using extraction heuristics (cf. (Koehn et al., 2003; Och and Ney, 2004)). These heuristics define a phrase pair to consist of a source and target ngrams of a word-aligned source-target sentence pair such that if one end of an alignment is in the one ngram, the other end is in the other ngram (and there is at least one such alignment) (Och and Ney, 2004; Koehn et al., 2003). For efficiency and sparseness, the practitioners of PBSMT constrain the length of the source phrase to a certain maximum number of words.

An All Phrase Pairs Model: In this work we train a phrase-translation table that consists of *all phrase-pairs* that can be extracted from the word-aligned training data according to the standard phrase extraction heuristic. After training, we can still limit the set of phrase pairs to those selected by a cut-off on phrase length. The reason for using all phrase pairs during training is that it gives a clear point of reference for an estimator, without implicit, accidental biases that might emerge due to length cut-off[1].

The Generative Model: Given a word-aligned source-target sentence pair $\langle \mathbf{f}, \mathbf{e}, \mathbf{a} \rangle$, the generative story underlying our model goes as follows:

1. Abiding by the word-alignments in $\mathbf{a}$, segment the source-target sentence pair $\langle \mathbf{f}, \mathbf{e} \rangle$ into a sequence of I containers σ_1^I, and a bag of I phrase pairs $\sigma_1^I(\mathbf{f}, \mathbf{e}) = \{\langle f_j, e_j \rangle\}_{j=1}^I$. Each container $\sigma_j = \langle l_f, r_f, l_e, r_e \rangle$ consists of the start l_f and end r_f positions[2] for a phrase in $\mathbf{f}$ and the start l_e and end r_e positions for an aligned phrase in $\mathbf{e}$.

2. For a given segmentation σ_1^I, for every container σ_j ($1 \leq j \leq I$) generate the phrase-pair $\langle f_j, e_j \rangle$, independently from all other phrase-pairs.

This leads to the following probabilistic model:

$$P(\mathbf{f} \mid \mathbf{e}; \mathbf{a}) = \sum_{\sigma_1^I \in \Sigma(\mathbf{a})} P(\sigma_1^I) \prod_{\langle f_j, e_j \rangle \in \sigma_1^I(\mathbf{f}, \mathbf{e})} P(f_j \mid e_j) \quad (1)$$

Where $\Sigma(\mathbf{a})$ is the set of *binarizable* segmentations (defined next) that are eligible according to the word-alignments $\mathbf{a}$ between $\mathbf{f}$ and $\mathbf{e}$. These segmentations into bilingual containers (where segmentations are taken inside the containers) are different from the monolingual segmentations used in earlier comparable conditional models (e.g., (DeNero et al., 2006)) which must generate the alignment on top of the segmentations. Note how the different phrase pairs $\langle f_j, e_j \rangle$ are generated from their bilingual containers in the given segmentation σ_1^I. We will discuss our choice of prior probability over segmentations $P(\sigma_1^I)$ after we discuss the definition of the binarizable segmentations $\Sigma(\mathbf{a})$.

3.1 Binarizable segmentations $\Sigma(\mathbf{a})$

Following (Zhang et al., 2006; Huang et al., 2008), every sequence of *phrase alignments* can be viewed

[1]For example, if the cut-off on phrase pairs is ten words, all sentence pairs smaller than ten words in the training data will be included as phrase pairs as well. These sentences are treated differently from longer sentences, which are not allowed to be phrase pairs.

[2]The NULL alignments (word-to-NULL) in the training data can also be marked with actual positions on both sides in order to allow for this definition of containers.

as a sequence of integers $1, \ldots I$ together with a permuted version of this sequence $\pi(1), \ldots, \pi(I)$, where the two copies of an integer in the two sequences are assumed aligned/paired together. For example, possible permutations of $\{1, 2, 3, 4\}$ are $\{2, 1, 3, 4\}$ and $\{2, 4, 1, 3\}$. Because a segmentation σ_1^I of a sentence pair is also a sequence of aligned phrases, it also constitutes a permuted sequence. A binarizable permutation $\mathbf{x}$ is either of length one, or can be **properly split** into two binarizable sub-sequences $\mathbf{y}$ and $\mathbf{z}$ such that either[3] $\mathbf{z} < y$ or $\mathbf{y} < z$. For example, one way to binarize the permutation $\{2, 1, 3, 4\}$ is to introduce a proper split into $\{2, 1; 3, 4\}$, then recursively another proper split of $\{2, 1\}$ into $\{2; 1\}$ and $\{3, 4\}$ into $\{3; 4\}$. In contrast, the permutation $\{2, 4, 1, 3\}$ is non-binarizable.

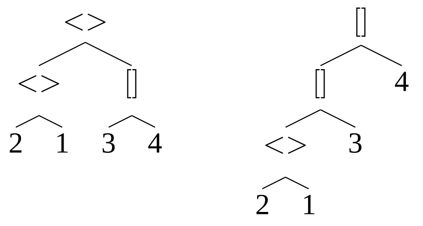


Figure 1: Multiple ways to binarize a permutation

Graphically speaking, the recursive definition of binarizable permutations can be depicted as a binary tree structure where the nodes correspond to recursive proper splits of the permutation, and the leaves are decorated with the naturals. Figure 1 exhibits two possible binarizations of the same permutation where <> and [] denote inverted and monotone proper splits respectively. Note that the number of possible binarizations of a binarizable permutation is a recursive function of the number of possible proper splits and reaches its maximum for fully monotone permutations (all binary trees, which is a factorial function of the length of the permutation).

By definition (cf. (Zhang et al., 2006; Huang et al., 2008)), a binarizable segmentation/permutation can be recognized by a binarized Synchronous Context-Free Grammar (SCFG), i.e., an SCFG in which the right hand sides of all non-lexical rules constitute binarizable permutations. In particular, this holds for the SCFG implementing Inversion Transduction Grammar (Wu, 1997). This SCFG (Chiang, 2005b) has two binary synchronous rules that correspond resp. to the contiguous monotone and inverted alignments:

$$\begin{aligned} XP &\rightarrow XP^{\boxed{1}}\, XP^{\boxed{2}},\ XP^{\boxed{1}}\, XP^{\boxed{2}} \\ XP &\rightarrow XP^{\boxed{1}}\, XP^{\boxed{2}},\ XP^{\boxed{2}}\, XP^{\boxed{1}} \end{aligned} \tag{2}$$

The boxed integers in the superscripts on the non-terminal XP denote synchronized rewritings. In this work, we employ a binary SCFG (bSCFG) working with these two synchronous rules together with a set of lexical rules $\{XP \rightarrow f,\ e \mid \langle f, e\rangle\ is\ a\ phrase\ pair\}$.

In this bSCFG, every derivation corresponds to a binarization of a segmentation of the input. Note that the bSCFG defined in equation 2 generates all possible binarizations for every segmentation of the input. It is possible to constrain this bSCFG such that it generates a single, canonical derivation per segmentation. However, in section 3.2 we show that the number of such derivations is a good measure of phrase pair productivity.

It is well known that there are alignments and segmentations that this bSCFG does not cover (see (Huang et al., 2008)). Recently, strong evidence emerged (e.g., (Huang et al., 2008)) showing that most word-alignments of actual parallel corpora can be covered by a binarized SCFG of the ITG type. Furthermore, because our model employs the set of *all phrase-pairs* that can be extracted from a given training set, it will always find segmentations that cover every sentence pair in the training data[4]. This implies that while our model might discard non-binarizable segmentations for certain complex word alignments, we do manage to train the model on the binarizable segementations of all sentence pairs.

Up to the prior over segmentations (see next), we implement the above model using a weighted version of the binary SCFG as follows:

- The weight for lexical rules is given by $P(XP \rightarrow f, e) := P(f \mid e)$, where $\langle f, e\rangle$ is a phrase-pair. These are the trainable parameters of our model.

[3]For two sequences of numbers, the notation $\mathbf{y} < z$ stands for $\forall y \in \mathbf{y}, \forall z \in \mathbf{z} : y < z$.

[4]In the worst case the whole sentence pair is a phrase pair with a trivial segmentation.

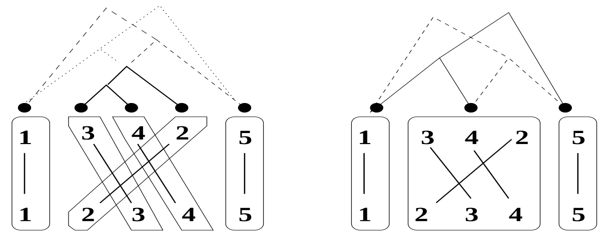


Figure 2: Two segmentations of an alignment/permutation. Both segmentations have the same number of binarizations despite differences in container sizes.

- The weights for the two non-lexical rules in equation 2 are fixed at 1.0. These weights are not trained at all.

Where we use the notation $P(.)$ for the weight of a synchronous rule.

3.2 Prior over segmentations

As it has been found out by (DeNero et al., 2006), it is not easy to come up with a simple, effective prior distribution over segmentations that allows for improved phrase pair estimates. Within a Maximum-Likelihood estimator, preference for segmentations σ_1^I consisting of longer containers could lead to overfitting as we will explain in section 4. Alternatively, it is tempting to have preference for segmentations σ_1^I that consist of shorter containers, because (generally speaking) shorter containers have higher expected coverage of new sentence pairs. However, mere bias for shorter containers will not give better estimates as observed by (DeNero et al., 2006). One case where this bias clearly fails is the case of a contiguous sequence of containers with a complex alignment structure (crossing alignments). For example (see figure 2), for the alignment $\{1, 3, 4, 2, 5\}$ there is a segmentation into five containers $\{1; 3; 4; 2; 5\}$, and another into three $\{1; 3, 4, 2; 5\}$. The first segmentation involves shorter containers that have crossing brackets among them, while the second one consists of three containers including a longer container $\{3, 4, 2\}$. In the first segmentation, due to their crossing alignments, each of the containers $\{3\}$, $\{4\}$ and $\{2\}$ will not combine with the surrounding context ($\{1\}$ and $\{5\}$) on its own, i.e., without the other two containers. Furthermore, there is only a single binarization of $\{3, 4, 2\}$. Hence, while the first segmentation involves shorter containers than the second one, these shorter containers are as *productive* as the large container $\{3, 4, 2\}$, i.e., they combine with surrounding containers in the same number of ways as the large container. In such and similar cases, there are no grounds for the bias towards shorter phrases/containers.

The notion of *container productivity* (the number of ways in which it combines with surrounding containers during training) seems to correlate with the expected number of ways a container can be used during decoding, which should be correlated with expected coverage. During training, containers that are often surrounded by other, monotonically aligned containers are expected to be more productive than alternative containers that are often surrounded by crossing alignments. Hence, the number of binarizations that a segmentation has under the bSCFG is a direct function of the ways in which the containers combine among themselves (monotone vs. inverted/crossing) within segmentations, and provides a more accurate measure of container productivity than container length. Hence, the final model we employ is the following:

$$P(\mathbf{f} \mid \mathbf{e}; \mathbf{a}) = \sum_{\sigma_1^I \in \Sigma(\mathbf{a})} \frac{N(\sigma_1^I)}{Z(\Sigma(\mathbf{a}))} \prod_{\langle f_j, e_j \rangle \in \sigma_1^I(\mathbf{f}, \mathbf{e})} P(f_j \mid e_j) \quad (3)$$

Where $N(\sigma_1^I)$ is the number of binary derivations/trees that σ_1^I has in the binary SCFG (bSCFG), and $Z(\Sigma(\mathbf{a})) = \sum_{\sigma_1^J \in \Sigma(\mathbf{a})} N(\sigma_1^J)$, i.e., this prior is the ratio of number of derivations of σ_1^I to the total number of derivations that $\langle \mathbf{f}, \mathbf{e}, \mathbf{a} \rangle$ has under the bSCFG.

3.3 Contrast with similar models:

In contrast with the model of (DeNero et al., 2006), who define the segmentations over the source sentence $\mathbf{f}$ alone, our model employs bilingual containers thereby segmenting both source and target sides simultaneously. Therefore, unlike (DeNero et al., 2006), our model does not need to generate the word-alignments explicitly, as these are embedded in the segmentations. Similarly, our model does not include *explicit* penalty terms for reordering/inversion but includes a related bias in the prior probabilities over segmentations $P(\sigma_1^I)$.

In a way, the segmentations and bilingual containers we use can be viewed as similar to the concepts used in the Joint Model of Marcu and Wong (Marcu and Wong, 2002). Unlike (Marcu and Wong, 2002), however, our model works with conditional probabilities and starts out from the word-alignments.

The novel aspects of our model are three (1) It defines the set of segmentations using a bSCFG, (2) It includes a novel, refined prior probability over segmentations, and (3) It employs all phrase pairs that can be extracted from a word-aligned training parallel corpus. For these novel elements to produce reasonable estimates, we devise our own estimator.

4 Estimation by Smoothing

In principle, we are dealing here with a translation model that employs all phrase pairs (of unbounded size), extracted from a word-aligned parallel corpus. Under this model, where a phrase pair and its sub-phrase pairs are included in the model, the MLE can be expected to overfit the data[5] unless a suitable prior probability over segmentations is employed. Indeed, the prior over segmentations defined in the preceding section prevents the MLE from completely overfitting the training data. However, we find empirical evidence that this prior is insufficient for avoiding overfitting.

Our model behaves like a *memory-based model* because it memorizes all extractable phrase pairs found in the training data including the training sentence pairs themselves. Such memory-based models are related to nonparametric models such as K-NN and kernel methods (Hastie et al., 2001). For memory-based models, consistent estimation for novel instances proceeds by local density estimation from the surroundings of the instance, which is akin to smoothing for parametric models. Hence, next we describe our own version of a *smoothed* Maximum-Likelihood estimator for phrase translation probabilities.

```
INPUT: Word-aligned parallel training data $T$
OUTPUT: Estimates $\pi$ for all $P(f \mid e)$
{
Split training data $T$ into equal parts $H_1, \ldots, H_{10}$.
For $1 \leq i \leq 10$ do
    Extract from $E_i = \cup_{j \neq i} H_j$ all phrase pairs $\pi_i$
    Initialize $\widehat{\pi}_i^0$ to uniform conditional probs
Let $j = 0$
Repeat
  Let $j = j + 1$        // EM iteration counter
  For $1 \leq i \leq 10$ do
      E-step: calculate expected counts for pairs
          in $\pi_i^j$ on $H_i$ using counts from $\widehat{\pi}_i^{j-1}$.
      M-step: calculate probabilities for pairs in
          $\pi_i^j$ from the expected counts
  For $1 \leq i \leq 10$ do $\widehat{\pi}_i^j := \frac{1}{10} \sum_{i=1}^{10} \pi_i^j$
Until $\pi := \{\widehat{\pi}_1^j, \ldots, \widehat{\pi}_{10}^j\}$ has converged
}
```

Figure 3: Penalized Deleted Estimation

For a latent variable model, it is usually common to employ Expectation-Maximization (EM) (Dempster et al., 1977) as a search method for a (local) maximum-likelihood estimate (MLE) of the training data. Instead of mere EM we opt for a *smoothed* version: we present a new method, that combines Deleted Estimation (Jelinek and Mercer, 1980) with the Jackknife (Duda et al., 2001) as the core estimator.

Figure 3 shows the pseudo-code for our estimator. Like in Deleted Estimation, we split the training data into ten equal portions. This way we create ten different splits of *extraction/heldout sets* of respectively 90%/10% of the training set. For every split $1 \leq i \leq 10$, we extract a set of phrase pairs π_i from the *extraction* set E_i and train it (under our model) on the *heldout set* H_i. Naturally, the phrase pair sets π_i ($1 \leq i \leq 10$) are subsets of (or equal to) the set of phrase pairs $\pi = \cup_i \pi_i$ extracted from the total training data (i.e., π is the set of model parameters). The training of the different π_i's, each on its corresponding heldout set H_i, is done by ten separate EM processes, which are synchronized in their initializa-

[5]One trivial MLE solution would give the longest container, consisting of the longest phrase pairs, a probability of one, at the cost of all shorter alternatives. A similar problem arises in Data-Oriented Parsing, see (Sima'an and Buratto, 2003; Zollmann and Sima'an, 2006). Note that models that employ an upperbound on phrase pair length will still risk overfitting training sentences of lengths that fall within this upperbound.

tion, their iterations as well as stop condition. The EM processes start out from uniform conditional estimates of the phrase pairs in all π_i. After every EM iteration j, when the M-step has finished, the estimates in all π_i^j ($1 \leq i \leq 10$) are set to the average (over $1 \leq i \leq 10$) of the estimates in π_i^j leading to $\widehat{\pi}_i^j$ (following the Jackknife method). The resulting averaged probabilities in $\widehat{\pi}_i^j$ are then used as the current phrase pair estimates, which feed into the next iteration $j+1$ of the different EM processes (each working on a different heldout set H_i with a different set of phrase pairs π_i).

There are two special boundary cases which demand special attention during estimation:

Sparse distributions: For a phrase e that does occur both in H_i and E_i, there could be a phrase pair $\langle f, e\rangle$ that does occur in H_i but does *not* occur in π_i. To prevent EM from giving the extra probability mass to all other pairs $\langle f, e^{'}\rangle$ unjustifiably, we apply smoothing. We add the missing pair $\langle f, e\rangle$ to π_i and set its probability to a fixed number 10^{-5*len}, where len is the length of the phrase pair. In effect, we backoff our model (equation 1) to a word-level model with fixed word translation probability (10^{-5}).

Zero distributions: When a phrase e does not occur in H_i, all its pairs $\langle f, e\rangle$ in π_i will have zero counts. During each EM iteration, when the M-step is applied, the distribution $P(\cdot \mid e)$ is undefined by MLE, since it is irrelevant for the likelihood of H_i. In this case any choice of proper distribution $P(\cdot \mid e)$ will constitute an MLE solution. We choose to set this case to a uniform distribution every time again.

Since our model and estimator are implemented within the bSCFG framework, we use a bilingual CYK parser (Younger, 1967) under the grammar in equation 2. This parser builds for every input $\langle \mathbf{f}, \mathbf{a}, \mathbf{e}\rangle$ all binarizations/derivations for every segmentation in $\Sigma(\mathbf{a})$. For implementing EM, we employ the Inside-Outside algorithm (Lari and Young, 1990; Goodman, 1998). During estimation, because the input, output and word-alignment are known in advance, the time and space requirements remain manageable despite the worst-case complexity $O(n^6)$ in target sentence length n.

Penalized Deleted Estimation: In contrast with our method, Deleted Estimation sums the *expected counts* (rather than probabilities) obtained from the different splits before applying the M-step (normalization). While the rationale behind Deleted Estimation comes from MLE over the original training data, our method has a smoothing objective (inspired by the Jackknife): generally speaking, the averages over different heldout sets (under different subsets of the model) give less sharp estimates than MLE. By averaging the different heldout estimates, this estimator employs a penalty term that depends on the marginal count of e in the heldout set[6]. Interestingly, when the phrase e is very frequent[7], it will approximately occur almost as often in the different heldout sets. In this case, our method reduces to Deleted Estimation, where it effectively sums the counts[8]. Yet, when the target phrase e does occur only very few times, it is likely that its count in some splits will be zero. In our method, at every EM iteration, during the Maximization step, we set such cases back to uniform. By averaging the probabilities from the different splits over many EM iterations, setting these cases to uniform constitutes a kind of prior that prevents the final estimates from falling too far from uniform. In contrast, in Deleted Interpolation the zero counts are simply summed with the other corresponding counts of the same phrase pair, which leads to sharper probability distributions. In all experiments that we conducted, our method (which we call *Penalized Deleted Estimation*) gave more successful estimates than mere Deleted Estimation.

On the theoretical side, the choice for a fixed

[6]Define $count_y(x)$ to be the count of event x in data y. The Deleted Estimation (DE) estimate is $\sum_H count_H(f,e)/count_T(e)$, which can be written as $\sum_H [count_H(f,e)/count_H(e)][count_H(e)/count_T(e)] = \sum_H \pi_H(f|e)[count_H(e)/count_T(e)]$ where $\pi_H(f|e)$ is the estimate from heldout set H. Hence, DE linearly interpolated π_H with factors $count_H(e)/count_T(e)$. Our estimator employs uniform interpolation factors instead, thereby penalizing the DI counts (hence Penalized DI).

[7]Theoretically speaking, when the training data is unboundedly large, our estimator will converge to the same estimates as the Deleted Estimation. When the data is still sparse, our estimator is biased, unlike the MLE which will overfit.

[8]When calculating the conditional probabilities, the denominators used are approximately equal to one another.

prior over segmentations (ITG prior) implies that our model cannot be estimated to converge (in probability) to the relative frequency estimates (RFE) of source-target sentence pairs in the limit of the training data (a sufficiently large parallel corpus). A prior probability over segmentations that would allow our estimator to converge in the limit to the RFE must gradually prefer segmentations consisting of larger containers as the data grows large. We set the design and estimation of such a prior aside for future work.

Phrases	**System**	**BLEU**
≤ 7	Baseline PBSMT	33.03
≤ 10	Baseline PBSMT	33.03
All	Baseline PBSMT	33.00
≤ 7	EM + ITG Prior	32.50
≤ 7	EM + Del. Est.	32.67
≤ 7	EM + Del. Est. + ITG Prior	32.73
≤ 7	EM + Pen. Del. Est. + ITG Prior	33.02
≤ 10	EM + Pen. Del. Est. + ITG Prior	**33.14**
All	EM + Pen. Del. Est. + ITG Prior	32.98

Table 1: Results: data from ACL07 2^{nd} Wkshp on SMT

5 Empirical experiments

Decoding and Baseline Model: In this work we employ an existing decoder, Moses (Hoang and Koehn, 2008), which defines a log-linear model interpolating feature functions, with interpolation scores λ_f $\mathbf{e}^* = \arg\max_{\mathbf{e}} \sum_{f \in \Phi} \lambda_f H_f(\mathbf{f}, \mathbf{e})$. The λ_f are optimized by Minimum-Error Training (MERT) (Och, 2003). The set Φ consists of the following feature functions (see (Hoang and Koehn, 2008)): a 5-gram target language model, the standard reordering scores, the word and phrase penalty scores, the conditional lexical estimates obtained from the word-alignment in both directions, and the conditional phrase translation estimates in both directions $P(f \mid e)$ and $P(e \mid f)$. Keeping the other five feature functions fixed, we compare our estimates of $P(f \mid e)$ and $P(e \mid f)$ (and the phrase penalty) to the commonly used heuristic estimates.

Because our model employs a latent segmentation variable, this variable should be marginalized out during decoding to allow selecting the highest probability translation given the input. This turns out crucial for improved results (cf. (Blunsom et al., 2008)). However, such a marginalization can be NP-Complete, in analogy to a similar problem in Data-Oriented Parsing (Sima'an, 2002)[9]. We do not have a decoder yet that can approximate this marginalization efficiently and we employ the standard Moses decoder for this work.

Experimental Setup: The training, development and test data all come from the French-English translation shared task of the ACL 2007 Second Workshop on Statistical Machine Translation [10]. After pruning sentence pairs with word length more than 40 on either side, we are left with 949K sentence pairs for training. The development and test data are composed of 2K sentence pairs each. All data sets are lower-cased.

For both the baseline system and our method, we produce word-level alignments for the parallel training corpus using GIZA++. We use 5 iterations of each IBM Model 1 and HMM alignment models, followed by 3 iterations of each Model 3 and Model 4. From this aligned training corpus, we extract the phrase pairs according to the heuristics in (Koehn et al., 2003). The baseline system extracts all phrase-pairs upto a certain maximum length on both sides and employs the heuristic estimator. The language model used in all systems is a 5-gram language model trained on the English side of the parallel corpus. Minimum-Error Rate Training (MERT) is applied on the development set to obtain optimal log-linear interpolation weights for all systems. Performance is measured by computing the BLEU scores (Papineni et al., 2002) of the system's translations, when compared against a single reference translation per sentence.

Results: We compare different versions of our system against the baseline system using the heuristic estimator. We observe the effects of the ITG prior in the translation model as well as the method of estimation (Deleted Estimation vs. Penalized Deleted Estimation).

Table 1 exhibits the BLEU scores for the sys-

[9]A reduction of simple instances of the first problem to instances of the latter problem should be possible.

[10]http://www.statmt.org/wmt07

tems. Our own system (with ITG prior and Penalized Deleted Estimation and maximum phrase-length ten words) scores (33.14), slightly outperforming the best baseline system (33.03). When using straight Deleted Estimation over EM, this leads to deterioration (32.73). When also the ITG prior is excluded (by having a single derivation per segmentation) this leads to further deterioration (32.67). By using mere EM with an ITG prior, performance goes down to 32.50, exhibiting the crucial role of the estimation by smoothing. Clearly, Penalized Deleted Estimation and the ITG prior are important for the improved phrase translation estimates.

As table 1 shows we also varied the phrase length cutoff (seven, ten or none=all phrase pairs). The length cutoff pertains to both sides of a phrase-pair. For our estimator, we always train all phrase pairs, applying the length cutoff only after training (no re-normalization is applied at that point).

Interestingly, we find out that the heuristic estimator cannot benefit performance by including longer phrase pairs. Our estimator does benefit performance by including phrase pairs of length upto ten words, but then it degrades again when including all phrase pairs. We take the latter finding to signal remaining overfitting that proved resistant to the smoothing applied by our estimator. The heuristic estimator exhibits a similar degradation.

We also tried to vary the treatment of Sparse Distributions (section 4, page 7) during heldout estimation from fixed word-translation probabilities to the lexical model probabilities. This lead to slight deterioration of results (32.94). It is unclear whether this deterioration is meaningful or not. We did not explore mere EM without any smoothing or ITG prior, as we expect it will directly overfit the training data as reported by (DeNero et al., 2006).

We note that for French-English translation it is hard to outperform the heuristic within the PBSMT framework, since it already performs very well. Preliminary, most recent experiments on German-English (also WMT07 data) exhibit that our estimator outperforms the heuristic.

6 Discussion and Future Research

The most similar efforts to ours, mainly (DeNero et al., 2006), conclude that segmentation variables in the generative translation model lead to overfitting while attaining higher likelihood of the training data than the heuristic estimator. Based on this advise (Moore and Quirk, 2007) exclude the latent segmentation variables and opt for a heuristic training procedure. In this work we also start out from a generative model with latent segmentation variables. However, we find out that concentrating the learning effort on smoothing is crucial for good performance. For this, we devise ITG-based priors over segmentations and employ a penalized version of Deleted Estimation working with EM at its core. The fact that our results (at least) match the heuristic estimates on a reasonably sized data set (947k parallel sentence pairs) is rather encouraging.

The work in (Zhang et al., 2008) has a similar flavor to our work, yet the two differ substantially. Both depart from Maximum-Likelihood towards non-overfitting estimators. Where Zhang et al choose for sparse priors (leading to sharp phrase distributions) and put the smoothing burden on the ITG rule parameters and a pruning strategy, we choose for a prior over segmentations determined by the ITG derivation space and smooth the MLE directly with a penalized version of Deleted Estimation. It remains to be seen how the two biases compare to one another on the same task.

There are various strands of future research. Firstly, we plan to explore our estimator on other language pairs in order to obtain more evidence on its behavior. Secondly, as (Blunsom et al., 2008) show, marginalizing out the different segmentations during decoding leads to improved performance. We plan to build our own decoder (based on ITG) where different ideas can be tested including tractable ways for achieving a marginalization effect. Apart from a new decoder, it will be worthwhile adapting the prior probability in our model to allow for consistent estimation. Finally, it would be interesting to study properties of the penalized Deleted Estimation used in this paper.

Acknowledgments: Both authors are supported by a VIDI grant (nr. 639.022.604) from The Netherlands Organization for Scientific Research (NWO). David Chiang and Andy Way are acknowledged for stimulating discussions on machine translation and parsing.

References

A. Birch, Ch. Callison-Burch, M. Osborne, and Ph. Koehn. 2006. Constraining the phrase-based, joint probability statistical translation model. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 154–157. Association for Computational Linguistics.

P. Blunsom, T. Cohn, and M. Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proceedings of ACL-08: HLT*, pages 200–208. Association for Computational Linguistics.

D. Chiang. 2005a. A hierarchical phrase-based model for statistical machine translation. In *In Proceedings of ACL 2005*, pages 263–270.

D. Chiang. 2005b. An introduction to synchronous grammars. Technical report, Univeristy of Maryland.

A.P. Dempster, N.M. Laird, and D.B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

J. DeNero, D. Gillick, J. Zhang, and D. Klein. 2006. Why generative phrase models underperform surface heuristics. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 31–38, New York City. Association for Computational Linguistics.

R.O. Duda, P.E. Hart, and D.G. Stork. 2001. *Pattern Classification*. John Wiley & Sons, NY, USA.

J.T. Goodman. 1998. *Parsing Inside-Out*. PhD thesis, Departement of Computer Science, Harvard University, Cambridge, Massachusetts.

T. Hastie, R. Tibshirani, and J. H. Friedman. 2001. *The Elements of Statistical Learning*. Springer.

H. Hoang and Ph. Koehn. 2008. Design of the moses decoder for statistical machine translation. In *ACL Workshop on Software engineering, testing, and quality assurance for NLP 2008*.

L. Huang, H. Zhang, D. Gildea, and K. Knight. 2008. Binarization of synchronous context-free grammars. *Submitted to Computational Linguistics. http://www.cis.upenn.edu/ lhuang3/opt.pdf*.

F. Jelinek and R. L. Mercer. 1980. Interpolated estimation of markov source parameters from sparse data. In *In Proceedings of the Workshop on Pattern Recognition in Practice*.

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

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.

D. Marcu and W. Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of Empirical methods in natural language processing*, pages 133–139. Association for Computational Linguistics.

R. Moore and Ch. Quirk. 2007. An iteratively-trained segmentation-free phrase translation model for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 112–119, Prague, Czech Republic. Association for Computational Linguistics.

F. J. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

F. J. Och and H. Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

F. J. Och. 2003. Minimum error rate training in statistical machine translation. In *ACL*, pages 160–167.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

K. Sima'an and L. Buratto. 2003. Backoff Parameter Estimation for the DOP Model. In H. Blockeel N. LavraĈ, D. Gamberger and L. Todorovski, editors, *Proceedings of the 14th European Conference on Machine Learning (ECML'03), Lecture Notes in Artificial Intelligence (LNAI 2837)*, pages 373–384, Cavtat-Dubrovnik, Croatia. Springer.

K. Sima'an. 2002. Computational complexity of probabilistic disambiguation. *Grammars*, 5(2):125–151.

D. Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

D.H. Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208.

R. Zens, F. J. Och, and H. Ney. 2002. Phrase-based statistical machine translation. In Matthias Jarke, Jana Koehler, and Gerhard Lakemeyer, editors, *KI 2002: Advances in Artificial Intelligence, 25th Annual German Conference on AI (KI 2002)*, volume 2479 of *Lecture Notes in Computer Science*, pages 18–32. Springer.

H. Zhang, L. Huang, D. Gildea, and K. Knight. 2006. Synchronous binarization for machine translation. In *HLT-NAACL*.

H. Zhang, Ch. Quirk, R. C. Moore, and D. Gildea. 2008. Bayesian learning of non-compositional phrases with synchronous parsing. In *Proceedings of ACL-08: HLT*, pages 97–105, Columbus, Ohio, June. Association for Computational Linguistics.

A. Zollmann and K. Sima'an. 2006. An efficient and consistent estimator for data-oriented parsing. *Journal of Automata, Languages and Combinatorics (JALC)*, 10 (2005) Number 2/3:367–388.

Unsupervised Models for Coreference Resolution

Vincent Ng
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
vince@hlt.utdallas.edu

Abstract

We present a generative model for unsupervised coreference resolution that views coreference as an EM clustering process. For comparison purposes, we revisit Haghighi and Klein's (2007) fully-generative Bayesian model for unsupervised coreference resolution, discuss its potential weaknesses and consequently propose three modifications to their model. Experimental results on the ACE data sets show that our model outperforms their original model by a large margin and compares favorably to the modified model.

1 Introduction

Coreference resolution is the problem of identifying which *mentions* (i.e., noun phrases) refer to which real-world *entities*. The availability of annotated coreference corpora produced as a result of the MUC conferences and the ACE evaluations has prompted the development of a variety of supervised machine learning approaches to coreference resolution in recent years. The focus of learning-based coreference research has also shifted from the acquisition of a *pairwise* model that determines whether two mentions are co-referring (e.g., Soon et al. (2001), Ng and Cardie (2002), Yang et al. (2003)) to the development of rich linguistic features (e.g., Ji et al. (2005), Ponzetto and Strube (2006)) and the exploitation of advanced techniques that involve joint learning (e.g., Daumé III and Marcu (2005)) and joint inference (e.g., Denis and Baldridge (2007)) for coreference resolution and a related extraction task. The rich features, coupled with the increased complexity of coreference models, have made these supervised approaches more dependent on labeled data and less applicable to languages for which little or no annotated data exists. Given the growing importance of multi-lingual processing in the NLP community, however, the development of unsupervised and weakly supervised approaches for the automatic processing of resource-scarce languages has become more important than ever.

In fact, several popular weakly supervised learning algorithms such as self-training, co-training (Blum and Mitchell, 1998), and EM (Dempster et al., 1977) have been applied to coreference resolution (Ng and Cardie, 2003) and the related task of pronoun resolution (Müller et al., 2002; Kehler et al., 2004; Cherry and Bergsma, 2005). Given a small number of coreference-annotated documents and a large number of unlabeled documents, these weakly supervised learners aim to incrementally augment the labeled data by iteratively training a classifier[1] on the labeled data and using it to label mention pairs randomly drawn from the unlabeled documents as COREFERENT or NOT COREFERENT. However, classifying mention pairs using such iterative approaches is undesirable for coreference resolution: since the non-coreferent mention pairs significantly outnumber their coreferent counterparts, the resulting classifiers generally have an increasing tendency to (mis)label a pair as non-coreferent as bootstrapping progresses (see Ng and Cardie (2003)).

Motivated in part by these results, we present a generative, unsupervised model for probabilistically

[1]For co-training, a pair of *view* classifiers are trained; and for EM, a generative model is trained instead.

inducing coreference *partitions* on unlabeled *documents*, rather than classifying mention pairs, via EM clustering (Section 2). In fact, our model combines the best of two worlds: it operates at the document level, while exploiting essential linguistic constraints on coreferent mentions (e.g., gender and number agreement) provided by traditional pairwise classification models.

For comparison purposes, we revisit a fully-generative Bayesian model for unsupervised coreference resolution recently introduced by Haghighi and Klein (2007), discuss its potential weaknesses and consequently propose three modifications to their model (Section 3). Experimental results on the ACE data sets show that our model outperforms their original model by a large margin and compares favorably to the modified model (Section 4).

2 Coreference as EM Clustering

In this section, we will explain how we recast unsupervised coreference resolution as EM clustering. We begin by introducing some of the definitions and notations that we will use in this paper.

2.1 Definitions and Notations

A *mention* can be a pronoun, a name (i.e., a proper noun), or a nominal (i.e., a common noun). An *entity* is a set of coreferent mentions. Given a document D consisting of n mentions, $m_1, \ldots, m_n$, we use $Pairs(D)$ to denote the set of $\binom{n}{2}$ mention pairs, $\{m_{ij} \mid 1 \leq i < j \leq n\}$, where m_{ij} is formed from mentions m_i and m_j. The *pairwise probability* formed from m_i and m_j refers to the probability that the pair m_{ij} is coreferent and is denoted as $P_{coref}(m_{ij})$. A *clustering* of n mentions is an n x n Boolean matrix C, where C_{ij} (the (i,j)-th entry of C) is 1 if and only if mentions m_i and m_j are coreferent. An entry in C is *relevant* if it corresponds to a mention pair in $Pairs(D)$. A *valid clustering* is a clustering in which the relevant entries satisfy the transitivity constraint. In other words, C is valid if and only if $(C_{ij} = 1 \wedge C_{jk} = 1) \Longrightarrow C_{ik} = 1$ $\forall\ 1 \leq i < j < k \leq n$. Hence, a valid clustering corresponds to a *partition* of a given set of mentions, and the goal of coreference resolution is to produce a valid clustering in which each cluster corresponds to a distinct entity.

2.2 The Model

As mentioned previously, our generative model operates at the document level, inducing a valid clustering on a given document D. More specifically, our model consists of two steps. It first chooses a clustering C based on some clustering distribution $P(C)$, and then generates D given C:

$$P(D, C) = P(C)P(D \mid C).$$

To facilitate the incorporation of linguistic constraints defined on a pair of mentions, we represent D by its mention pairs, $Pairs(D)$. Now, assuming that these mention pairs are generated conditionally independently of each other given C_{ij},

$$P(D \mid C) = \prod_{m_{ij} \in Pairs(D)} P(m_{ij} \mid C_{ij}).$$

Next, we represent m_{ij} as a set of seven features that is potentially useful for determining whether m_i and m_j are coreferent (see Table 1).[2] Hence, we can rewrite $P(D \mid C)$ as

$$\prod_{m_{ij} \in Pairs(D)} P(m_{ij}^1, \ldots, m_{ij}^7 \mid C_{ij}),$$

where m_{ij}^k is the value of the kth feature of m_{ij}.

To reduce data sparseness and improve the estimation of the above probabilities, we make conditional independence assumptions about the generation of these feature values. Specifically, as shown in the first column of Table 1, we divide the seven features into three *groups* (namely, strong coreference indicators, linguistic constraints, and mention types), assuming that two feature values are conditionally independent if and only if the corresponding features belong to different groups. With this assumption, we can decompose $P(m_{ij}^1, \ldots m_{ij}^7 \mid C_{ij})$ into a product of three probabilities: $P(m_{ij}^1, m_{ij}^2, m_{ij}^3 \mid C_{ij})$, $P(m_{ij}^4, m_{ij}^5, m_{ij}^6 \mid C_{ij})$, and $P(m_{ij}^7 \mid C_{ij})$. Each of these distributions represents a pair of multinomial distributions, one for the coreferent mention pairs (C_{ij} = 1) and the other for the non-coreferent mention pairs (C_{ij} = 0). Hence, the set of parameters of our model, Θ, consists of $P(m^1, m^2, m^3 \mid c)$, $P(m^4, m^5, m^6 \mid c)$, and $P(m^7 \mid c)$.

[2]See Soon et al. (2001) for details on feature value computations. Note that all feature values are computed automatically.

Feature Type	Feature ID	Feature	Description
Strong Coreference Indicators	1	STR_MATCH	T if neither of the two mentions is a pronoun and after discarding determiners, the string denoting mention m_i is identical to that of mention m_j; else F.
	2	ALIAS	T if one mention is an acronym, an abbreviation, or a name variant of the other; else F. For instance, *Bill Clinton* and *President Clinton* are aliases, so are *MIT* and *Massachusetts Institute of Technology*.
	3	APPOSITIVE	T if the mentions are in an appositive relationship; else F.
Linguistic Constraints	4	GENDER	T if the mentions agree in gender; F if they disagree; NA if gender information for one or both mentions cannot be determined.
	5	NUMBER	T if the mentions agree in number; F if they disagree; NA if number information for one or both mentions cannot be determined.
	6	SEM_CLASS	T if the mentions have the same semantic class; F if they don't; NA if the semantic class information for one or both mentions cannot be determined.
Mention Types	7	NPTYPE	the feature value is the concatenation of the mention type of the two mentions, $t_i t_j$, where $t_i, t_j \in \{$ PRONOUN, NAME, NOMINAL $\}$.

Table 1: Feature set for representing a mention pair. The first six features are relational features that test whether some property P holds for the mention pair under consideration and indicate whether the mention pair is **T**RUE or **F**ALSE w.r.t. P; a value of **N**OT **A**PPLICABLE is used when property P does not apply.

2.3 The Induction Algorithm

To induce a clustering C on a document D, we run EM on our model, treating D as observed data and C as hidden data. Specifically, we use EM to iteratively estimate the model parameters, Θ, from documents that are probabilistically labeled (with clusterings) and apply the resulting model to probabilistically re-label a document (with clusterings). More formally, we employ the following EM algorithm:

E-step: Compute the posterior probabilities of the clusterings, $P(C|D, \Theta)$, based on the current Θ.

M-step: Using $P(C|D, \Theta)$ computed in the E-step, find the Θ' that maximizes the expected complete log likelihood, $\sum_C P(C|D, \Theta) \log P(D, C|\Theta')$.

We begin the induction process at the M-step.[3] To find the Θ that maximizes the expected complete log likelihood, we use maximum likelihood estimation with add-one smoothing. Since $P(C|D, \Theta)$ is not available in the first EM iteration, we instead use an initial distribution over clusterings, $P(C)$. The question, then, is: which $P(C)$ should we use? One possibility is the uniform distribution over all (possibly invalid) clusterings. Another, presumably better, choice is a distribution that assigns non-zero probability mass to only the valid clusterings. Yet another possibility is to set $P(C)$ based on a document labeled with coreference information. In our experiments, we employ this last method, assigning a probability of one to the correct clustering of the labeled document (see Section 4.1 for details).

After (re-)estimating Θ in the M-step, we proceed to the E-step, where the goal is to find the conditional clustering probabilities. Given a document D, the number of coreference clusterings is exponential in the number of mentions in D, even if we limit our attention to those that are valid. To cope with this computational complexity, we approximate the E-step by computing only the conditional probabilities that correspond to the N most probable coreference clusterings given the current Θ. We identify the N most probable clusterings and compute their probabilities as follows. First, using the current Θ, we reverse the generative model and compute $P_{coref}(m_{ij})$ for each mention pair m_{ij} in $Pairs(D)$. Next, using these pairwise probabilities, we apply Luo et al.'s (2004) Bell tree approach to coreference resolution to compute the N-best clusterings and their probabilities (see Section 2.4 for details). Finally, to obtain the required conditional clustering probabilities for the E-step, we normalize the probabilities assigned to the N-best clusterings so that they sum to one.

2.4 Computing the N-Best Partitions

As described above, given the pairwise probabilities, we use Luo et al.'s (2004) algorithm to heuristically compute the N-best clusterings (or, more precisely, N-best partitions[4]) and their probabilities based on

[3]Another possibility, of course, is to begin at the E-step by making an initial guess at Θ.

[4]Note that Luo et al.'s search algorithm only produces valid clusterings, implying that the resulting N-best clusterings are

```
Input: $M = \{m_1, ..., m_n\}$: mentions, $N$: no. of best partitions
Output: $N$-best partitions
 1: // initialize the data structures that store partial partitions
 2: $H_1 := \{PP := \{[m_1]\}\}$, $S(PP) = 1$
 3: $H_2, ..., H_n = \emptyset$
 4: for $i$ = 2 to $n$
 5:    // process each partial partition
 6:    foreach $PP \in H_{i-1}$
 7:       // process each cluster in PP
 8:       foreach $C \in PP$
 9:          Extend $PP$ to $PP'$ by linking $m_i$ to $C$
10:          Compute $S(PP')$
11:          $H_i := H_i \cup \{PP'\}$
12:       Extend $PP$ to $PP^\delta$ by putting $m_i$ into a new cluster
13:       Compute $S(PP^\delta)$
14:       $H_i := H_i \cup \{PP^\delta\}$
15: return $N$ most probable partitions in $H_n$
```

Figure 1: Our implementation of Luo et al.'s algorithm

the Bell tree. Informally, each node in a Bell tree corresponds to an ith-order *partial* partition (i.e., a partition of the first i mentions of the given document), and the ith level of the tree contains *all* possible ith-order partial partitions. Hence, the set of leaf nodes constitutes all possible partitions of all of the mentions. The search for the N most probable partitions starts at the root, and a partitioning of the mentions is incrementally constructed as we move down the tree. Since an exhaustive search is computationally infeasible, Luo et al. employ a beam search procedure to explore only the most probable paths at each step of the search process. Figure 1 shows our implementation of this heuristic search algorithm.

The algorithm takes as input a set of n mentions (and their pairwise probabilities), and returns the N most probable partitionings of the mentions. It uses data structures S and the H_i's to store intermediate results. Specifically, $S(PP)$ stores the score of the partial partition PP. H_i is associated with the ith level of the Bell tree, and is used to store the most probable ith-order partial partitions. Each H_i has a maximum size of $2N$: if more than $2N$ partitions are inserted into a given H_i, then only the $2N$ most probable ones will be stored. This amounts to pruning the search space by employing a beam size of $2N$ (i.e., expanding only the $2N$ most probable partial partitions) at each step of the search.

The algorithm begins by initializing H_1 with the only partial partition of order one, $\{[m_1]\}$, which has a score of one (line 2). Then it processes the mentions sequentially, starting with m_2 (line 4). When processing m_i, it takes each partial partition PP in H_{i-1} and creates a set of ith-order partitions by extending PP with m_i in all possible ways. Specifically, for each cluster C (formed by a subset of the first i–1 mentions) in PP, the algorithm generates a new ith-order partition, PP', by linking m_i to C (line 9), and stores PP' in H_i (line 11). The score of PP', $S(PP')$, is computed by using the pairwise coreference probabilities as follows:

$$S(PP') = S(PP) \cdot \max_{m_k \in C} P_{coref}(m_{ki}).$$

Of course, PP can also be extended by putting m_i into a new cluster (line 12). This yields PP^δ, another partition to be inserted into H_i (line 14), and

$$S(PP^\delta) = \delta \cdot S(PP) \cdot (1 - \max_{k \in \{1,\ldots,i-1\}} P_{coref}(m_{ki})),$$

where δ (the *start penalty*) is a positive constant ($<$ 1) used to penalize partitions that start a new cluster. After processing each of the n mentions using the above steps, the algorithm returns the N most probable partitions in H_n (line 15).

Our implementation of Luo et al.'s search algorithm differs from their original algorithm only in terms of the number of pruning strategies adopted. Specifically, Luo et al. introduce a number of heuristics to prune the search space in order to speed up the search. We employ only the beam search heuristic, with a beam size that is five times larger than theirs. Our larger beam size, together with the fact that we do not use other pruning strategies, implies that we are searching through a larger part of the space than them, thus potentially yielding better partitions.

3 Haghighi and Klein's Coreference Model

To gauge the performance of our model, we compare it with a Bayesian model for unsupervised coreference resolution that was recently proposed by Haghighi and Klein (2007). In this section, we will give an overview of their model, discuss its weaknesses and propose three modifications to the model.

3.1 Notations

For consistency, we follow Haghighi and Klein's (H&K) notations. **Z** is the set of random variables

indeed partitions. This is desirable, as there is no reason for us to put non-zero probability mass on invalid clusterings.

that refer to (indices of) entities. ϕ_z is the set of parameters associated with entity z. ϕ is the entire set of model parameters, which includes all the ϕ_z's. Finally, $\mathbf{X}$ is the set of observed variables (e.g., the head of a mention). Given a document, the goal is to find the most probable assignment of entity indices to its mentions given the observed values. In other words, we want to maximize $P(\mathbf{Z}|\mathbf{X})$. In a Bayesian approach, we compute this probability by integrating out all the parameters. Specifically,

$$P(\mathbf{Z}|\mathbf{X}) = \int P(\mathbf{Z}|\mathbf{X}, \phi)P(\phi|\mathbf{X})d\phi.$$

3.2 The Original H&K Model

The original H&K model is composed of a set of models: the *basic* model and two other models (namely, the *pronoun head* model and the *salience* model) that aim to improve the basic model.[5]

3.2.1 Basic Model

The basic model generates a mention in a two-step process. First, an entity index is chosen according to an *entity distribution*, and then the head of the mention is generated given the entity index based on an entity-specific *head distribution*. Here, we assume that (1) all heads H are observed and (2) a mention is represented solely by its head noun, so nothing other than the head is generated. Furthermore, we assume that the head distribution is drawn from a symmetric Dirichlet with concentration λ_H. Hence,

$$P(H_{i,j} = h|\mathbf{Z}, \mathbf{H}^{-i,j}) \propto n_{h,z} + \lambda_H$$

where $H_{i,j}$ is the head of mention j in document i, and $n_{h,z}$ is the number of times head h is emitted by entity index z in $(\mathbf{Z}, \mathbf{H}^{-i,j})$.[6] On the other hand, since the number of entities in a document is *not* known a priori, we draw the entity distribution from a Dirichlet *process* with concentration α, effectively yielding a model with an infinite number of mixture components. Using the Chinese restaurant process representation (see Teh et al. (2006)),

$$P(Z_{ij} = z|\mathbf{Z}^{-i,j}) \propto \begin{cases} \alpha & , \text{ if } z = z_{new} \\ n_z & , \text{ otherwise} \end{cases}$$

where n_z is the number of mentions in $\mathbf{Z}^{-i,j}$ labeled with entity index z, and z_{new} is a new entity index not already in $\mathbf{Z}^{-i,j}$. To perform inference, we use Gibbs sampling (Geman and Geman, 1984) to generate samples from this conditional distribution:

$$P(Z_{i,j}|\mathbf{Z}^{-i,j}, \mathbf{H}) \propto P(Z_{i,j}|\mathbf{Z}^{-i,j})P(H_{i,j}|\mathbf{Z}, \mathbf{H}^{-i,j})$$

where the two distributions on the right are defined as above. Starting with a random assignment of entity indices to mentions, the Gibbs sampler iteratively re-samples an entity index according to this posterior distribution given the current assignment.

3.2.2 Pronoun Head Model

Head generation in the basic model is too simplistic: it has a strong tendency to assign the same entity index to mentions having the same head. This is particularly inappropriate for pronouns. Hence, we need a different model for generating pronouns.

Before introducing this pronoun head model, we need to augment the set of entity-specific parameters, which currently contains only a distribution over heads (ϕ^h_Z). Specifically, we add distributions ϕ^t_Z, ϕ^g_Z, and ϕ^n_Z over entity *properties*: ϕ^t_Z is a distribution over semantic types (PER, ORG, LOC, MISC), ϕ^g_Z over gender (MALE, FEMALE, EITHER, NEUTER), and ϕ^n_Z over number (SG, PL). We assume that each of these distributions is drawn from a symmetric Dirichlet. A small concentration parameter is used, since each entity should have a dominating value for each of these properties.

Now, to estimate ϕ^t_Z, ϕ^g_Z, and ϕ^n_Z, we need to know the gender, number, and semantic type of each mention. For some mentions (e.g., "he"), these properties are easy to compute; for others (e.g., "it"), they are not. Whenever a mention has unobserved properties, we need to fill in the missing values. We could resort to sampling, but sampling these properties is fairly inefficient. So, following H&K, we keep soft counts for each of these properties and use them rather than perform hard sampling.

When an entity z generates a pronoun h using the pronoun head model,[7] it first generates a gender g, a number n, and a semantic type t independently from the distributions ϕ^g_z, ϕ^n_z, and ϕ^t_z; and then generates h using the distribution $P(H = h|G = g, N =$

[5]H&K also present a cross-document coreference model, but since it focuses primarily on cross-document coreference and improves within-document coreference performance by only 1.5% in F-score, we will not consider this model here.

[6]$\mathbf{H}^{-i,j}$ is used as a shorthand for $\mathbf{H} - \{H_{i,j}\}$.

[7]While pronouns are generated by this pronoun head model, names and nominals continue to be handled by the basic model.

$n, T = t, \theta)$. Note that this last distribution is a global distribution that is independent of the chosen entity index. θ is a parameter drawn from a symmetric Dirichlet (with concentration λ_P) that encodes our prior knowledge of the relationship between a semantic type and a pronoun. For instance, given the type PERSON, there is a higher probability of generating "he" than "it". As a result, we maintain a list of compatible semantic types for each pronoun, and give a pronoun a count of (1 + λ_P) if it is compatible with the drawn semantic type; otherwise, we give it a count of λ_P. In essence, we use this prior to prefer the generation of pronouns that are compatible with the chosen semantic type.

3.2.3 Salience Model

Pronouns typically refer to salient entities, so the basic model could be improved by incorporating salience. We start by assuming that each entity has an activity score that is initially set to zero. Given a set of mentions and an assignment of entity indices to mentions, $\mathbf{Z}$, we process the mentions in a left-to-right manner. When a mention, m, is encountered, we multiply the activity score of each entity by 0.5 and add one to the activity score of the entity to which m belongs. This captures the intuitive notion that frequency and recency both play a role in determining salience. Next, we rank the entities based on their activity scores and discretize the ranks into five "salience" buckets S: TOP (1), HIGH (2–3), MID (4–6), LOW (7+), and NONE. Finally, this salience information is used to modify the entity distribution:[8]

$$P(Z_{ij} = z|\mathbf{Z}^{-i,j}) \propto n_z \cdot P(M_{i,j}|S_{i,j}, \mathbf{Z})$$

where $S_{i,j}$ is the salience value of the jth mention in document i, and $M_{i,j}$ is its mention type, which can take on one of three values: pronoun, name, and nominal. $P(M_{i,j}|S_{i,j}, \mathbf{Z})$, the distribution of mention type given salience, was computed from H&K's development corpus (see Table 2). According to the table, pronouns are preferred for salient entities, whereas names and nominals are preferred for entities that are less active.

[8]Rather than having just one probability term on the right hand side of the sampling equation, H&K actually have a product of probability terms, one for each mention that appears later than mention j in the given document. However, they acknowledge that having the product makes sampling inefficient, and decided to simplify the equation to this form in their evaluation.

Salience Feature	Pronoun	Name	Nominal
TOP	0.75	0.17	0.08
HIGH	0.55	0.28	0.17
MID	0.39	0.40	0.21
LOW	0.20	0.45	0.35
NONE	0.00	0.88	0.12

Table 2: Posterior distribution of mention type given salience (taken from Haghighi and Klein (2007))

3.3 Modifications to the H&K Model

Next, we discuss the potential weaknesses of H&K's model and propose three modifications to it.

Relaxed head generation. The basic model focuses on head matching, and is therefore likely to (incorrectly) posit *the large airport* and *the small airport* as coreferent, for instance. In fact, head matching is a relatively inaccurate indicator of coreference, in comparison to the "strong coreference indicators" shown in the first three rows of Table 1. To improve H&K's model, we replace head matching with these three strong indicators as follows. Given a document, we assign each of its mentions a *head index*, such that two mentions have the same head index if and only if at least one of the three strong indicators returns a value of True. Now, instead of generating a head, the head model generates a head index, thus increasing the likelihood that aliases are assigned the same entity index, for instance. Note that this modification is applied only to the basic model. In particular, pronoun generation continues to be handled by the pronoun head model and will not be affected. We hypothesize that this modification would improve precision, as the strong indicators are presumably more precise than head match.

Agreement constraints. While the pronoun head model naturally prefers that a pronoun be generated by an entity whose gender and number are compatible with those of the pronoun, the entity (index) that is re-sampled for a pronoun according to the sampling equation for $P(Z_{i,j}|\mathbf{Z}^{-i,j}, \mathbf{H})$ may still not be compatible with the pronoun with respect to gender and number. The reason is that an entity index is assigned based not only on the head distribution but also on the entity distribution. Since entities with many mentions are preferable to those with few mentions, it is possible for the model to favor the assignment of a grammatically incompatible entity (index) to a pronoun if the entity is sufficiently large. To eliminate this possibility, we enforce the agreement constraints at the global level. Specifically, we sample an entity index for a given mention with a non-zero probability if and only if the corresponding entity and the head of the mention agree in gender and number. We hypothesize that this modification would improve precision.

Pronoun-only salience. In Section 3.2.3, we motivate the need for salience using pronouns only, since proper names can to a large extent be resolved using string-matching facilities and are not particularly sensitive to salience. Nominals (especially definite descriptions), though more sensitive to salience than names, can also be resolved by simple string-matching heuristics in many cases (Vieira and Poesio, 2000; Strube et al., 2002). Hence, we hypothesize that the use of salience for names and nominals would adversely affect their resolution performance, as incorporating salience could diminish the role of string match in the resolution process, according to the sampling equations. Consequently, we modify H&K's model by limiting the application of salience to the resolution of pronouns only. We hypothesize that this change would improve precision.

4 Evaluation

4.1 Experimental Setup

To evaluate our EM-based model and H&K's model, we use the ACE 2003 coreference corpus, which is composed of three sections: Broadcast News (BNEWS), Newswire (NWIRE), and Newspaper (NPAPER). Each section is in turn composed of a training set and a test set. Due to space limitations, we will present evaluation results only for the test sets of BNEWS and NWIRE, but verified that the same performance trends can be observed on NPAPER as well. Unlike H&K, who report results using only true mentions (extracted from the answer keys), we show results for true mentions as well as system mentions that were extracted by an in-house noun phrase chunker. The relevant statistics of the BNEWS and NWIRE test sets are shown in Table 3.

	BNEWS	NWIRE
Number of documents	51	29
Number of true mentions	2608	2630
Number of system mentions	5424	5197

Table 3: Statistics of the BNEWS and NWIRE test sets

Scoring programs. To score the output of the coreference models, we employ the commonly-used MUC scoring program (Vilain et al., 1995) and the recently-developed CEAF scoring program (Luo, 2005). In the MUC scorer, recall is computed as the percentage of coreference *links* in the reference partition that appear in the system partition; precision is computed in the same fashion as recall, except that the roles of the reference partition and the system partition are reversed. As a *link-based* scoring program, the MUC scorer (1) does not reward successful identification of singleton entities and (2) tends to under-penalize partitions that have too few entities. The *entity-based* CEAF scorer was proposed in response to these two weaknesses. Specifically, it operates by computing the optimal alignment between the set of reference entities and the set of system entities. CEAF precision and recall are both positively correlated with the score of this optimal alignment, which is computed by summing over each aligned entity pair the number of mentions that appear in both entities of that pair. As a consequence, a system that proposes too many entities or too few entities will have low precision and recall.

Parameter initialization. We use a small amount of labeled data for parameter initialization for the two models. Specifically, for evaluations on the BNEWS test data, we use as labeled data one randomly-chosen document from the BNEWS training set, which has 58 true mentions and 102 system mentions. Similarly for NWIRE, where the chosen document has 42 true mentions and 72 system mentions. For our model, we use the labeled document to initialize the parameters. Also, we set N (the number of most probable partitions) to 50 and δ (the start penalty used in the Bell tree) to 0.8, the latter being recommended by Luo et al. (2004).

For H&K's model, we use the labeled data to tune the concentration parameter α. While H&K set α to 0.4 without much explanation, a moment's thought reveals that the choice of α should reflect the fraction of mentions that appear in a singleton cluster. We therefore estimate this value from the labeled document, yielding 0.4 for true mentions (which is consistent with H&K's choice) and 0.7 for system mentions. The remaining parameters, the λ's, are all set to e^{-4}, following H&K. In addition, as is commonly done in Bayesian approaches, we do not sample entities directly from the conditional distribution $P(\mathbf{Z}|\mathbf{X})$; rather, we sample from this distribution raised to the power $\exp \frac{ci}{k-1}$, where c=1.5, i is the current iteration number that starts at 0, and k (the number of sampling iterations) is set to 20. Finally, due to sampling and the fact that the initial assignment of entity indices to mentions is random, all the reported results for H&K's model are averaged over five runs.

4.2 Results and Discussions

The Heuristic baseline. As our first baseline, we employ a simple rule-based system that posits two mentions as coreferent if and only if at least one of the three strong coreference indicators listed in Table 1 returns True. Results of this baseline, reported in terms of recall (R), precision (P), and F-score (F) using the MUC scorer and the CEAF scorer, are shown in row 1 of Tables 4 and 5, respectively. Each row in these tables shows performance using true mentions and system mentions for the BNEWS and NWIRE data sets. As we can see, (1) recall is generally low, since this simple heuristic can only identify a small fraction of the coreference relations; (2) CEAF recall is consistently higher than MUC recall, since CEAF also rewards successful identification of non-coreference relations; and (3) precision for true mentions is higher than that for system mentions, since the number of non-coreferent pairs that satisfy the heuristic is larger for system mentions.

The Degenerate EM baseline. Our second baseline is obtained by running only one iteration of our EM-based coreference model. Specifically, it starts with the M-step by initializing the model parameters using the labeled document, and ends with the E-step by applying the resulting model (in combination with the Bell tree search algorithm) to obtain the most probable coreference partition for each test document. Since there is no parameter re-estimation, this baseline is effectively a purely supervised system trained on one (labeled) document.

Results are shown in row 2 of Tables 4 and 5. As we can see, recall is consistently much higher than precision, suggesting that the model has produced fewer entities than it should. Perhaps more interestingly, in comparison to the Heuristic baseline, Degenerate EM performs consistently worse according to CEAF but generally better according to MUC. This discrepancy stems from the aforementioned properties that MUC under-penalizes partitions with too few entities, whereas CEAF lowers both recall and precision when given such partitions.

Our EM-based coreference model. Our model operates in the same way as the Degenerate EM baseline, except that EM is run until convergence, with the test set being used as unlabeled data for parameter re-estimation. Any performance difference between our model and Degenerate EM can thus be attributed to EM's exploitation of the unlabeled data.

Results of our model are shown in row 3 of Tables 4 and 5. In comparison to Degenerate EM, MUC F-score increases by 4-5% for BNEWS and 4-21% for NWIRE; CEAF F-score increases even more dramatically, by 10-17% for BNEWS and 16-27% for NWIRE. Improvements stem primarily from large gains in precision and comparatively smaller loss in recall. Such improvements suggest that our model has effectively exploited the unlabeled data.

In comparison to the Heuristic baseline, we generally see increases in both recall and precision when system mentions are used, and as a result, F-score improves substantially by 7-15%. When true mentions are used, we still see gains in recall, but these gains are accompanied by loss in precision. F-score generally increases (by 2-22%), except for the case with NWIRE where we see a 0.5% drop in CEAF F-score as a result of a larger decrease in precision.

The Original H&K model. We use as our third baseline the Original H&K model (see Section 3.2). Results of this model are shown in row 4 of Tables 4 and 5.[9] Overall, it underperforms our model by 6-16% in MUC F-score and 6-14% in CEAF F-score, due primarily to considerable drop in both recall and precision in almost all cases.

The Modified H&K model. Next, we incorporate our three modifications into the Original H&K baseline one after the other. Results are shown in rows 5-7 of Tables 4 and 5. Several points deserve mentioning. First, the addition of each modification improves the F-score for both true and system mentions

[9]The H&K results shown here are not directly comparable with those reported in Haghighi and Klein (2007), since H&K evaluated their system on the ACE 2004 coreference corpus.

		Broadcast News (BNEWS)						Newswire (NWIRE)					
		True Mentions			System Mentions			True Mentions			System Mentions		
	Experiments	R	P	F	R	P	F	R	P	F	R	P	F
1	Heuristic Baseline	27.8	72.0	40.1	30.9	44.3	36.4	31.2	70.3	43.3	36.3	53.4	43.2
2	Degenerate EM Baseline	63.6	53.1	57.9	70.8	36.3	48.0	64.5	42.6	51.3	69.0	25.1	36.8
3	Our EM-based Model	56.1	71.4	**62.8**	42.4	66.0	51.6	47.0	68.3	**55.7**	55.2	60.6	**57.8**
4	Haghighi and Klein Baseline	49.4	60.2	54.3	50.8	40.7	45.2	44.7	55.5	49.5	43.0	40.9	41.9
5	+ Relaxed Head Generation	53.0	65.4	58.6	48.3	45.7	47.0	45.1	62.5	52.4	40.9	50.0	45.0
6	+ Agreement Constraints	53.6	68.7	60.2	50.4	47.5	48.9	44.6	63.7	52.5	41.7	51.2	46.0
7	+ Pronoun-only Salience	56.8	68.3	62.0	52.2	53.0	**52.6**	46.8	66.2	54.8	44.3	57.3	50.0
8	Fully Supervised Model	53.7	70.8	61.1	53.0	70.3	60.4	52.0	69.6	59.6	53.1	70.5	60.6

Table 4: Results obtained using the MUC scoring program for the Broadcast News and Newswire data sets

		Broadcast News (BNEWS)						Newswire (NWIRE)					
		True Mentions			System Mentions			True Mentions			System Mentions		
	Experiments	R	P	F	R	P	F	R	P	F	R	P	F
1	Heuristic Baseline	42.1	75.8	54.1	44.2	48.7	46.3	43.9	73.4	54.9	47.5	53.4	50.3
2	Degenerate EM Baseline	51.2	43.1	46.8	53.7	26.8	35.8	51.0	30.5	38.2	45.1	18.6	26.3
3	Our EM-based Model	53.3	60.5	**56.7**	47.5	59.6	**52.9**	49.2	60.7	54.4	53.5	52.1	**52.8**
4	Haghighi and Klein Baseline	43.7	48.8	46.1	46.0	33.9	39.0	45.5	51.7	48.4	44.6	39.2	41.7
5	+ Relaxed Head Generation	45.8	52.4	48.9	45.4	39.6	42.3	46.0	57.0	50.9	44.5	48.3	46.3
6	+ Agreement Constraints	51.8	60.5	55.8	50.6	43.8	47.0	47.8	60.1	53.2	46.5	50.4	48.4
7	+ Pronoun-only Salience	53.9	59.9	**56.7**	52.3	49.9	51.1	49.6	62.8	**55.4**	47.4	55.7	51.2
8	Fully Supervised Model	55.0	63.3	58.8	56.2	64.2	59.9	54.7	64.7	59.3	56.5	65.4	60.6

Table 5: Results obtained using the CEAF scoring program for the Broadcast News and Newswire data sets

in both data sets using both scorers. These results provide suggestive evidence that our modifications are highly beneficial. The three modifications, when applied in combination, improve Original H&K substantially by 5-8% in MUC F-score and 7-12% in CEAF F-score, yielding results that compare favorably to those of our model in almost all cases.

Second, the use of agreement constraints yields larger improvements with CEAF than with MUC. This discrepancy can be attributed to the fact that CEAF rewards the correct identification of non-coreference relations, whereas MUC does not. Since agreement constraints are intended primarily for disallowing coreference, they contribute to the successful identification of non-coreference relations and as a result yield gains in CEAF recall and precision.

Third, the results are largely consistent with our hypothesis that these modifications enhance precision. Together, they improve the precision of the Original H&K baseline by 8-16% (MUC) and 11-16% (CEAF), yielding a coreference model that compares favorably with our EM-based approach.

Comparison with a supervised model. Finally, we compare our EM-based model with a fully supervised coreference resolver. Inspired by state-of-the-art resolvers, we create our supervised classification model by training a discriminative learner (the C4.5 decision tree induction system (Quinlan, 1993)) with a diverse set of features (the 34 features described in Ng (2007)) on a large training set (the entire ACE 2003 coreference training corpus), and cluster using the Bell tree search algorithm. The fully supervised results shown in row 8 of Tables 4 and 5 suggest that our EM-based model has room for improvements, especially when system mentions are used.

5 Conclusions

We have presented a generative model for unsupervised coreference resolution that views coreference as an EM clustering process. Experimental results indicate that our model outperforms Haghighi and Klein's (2007) coreference model by a large margin on the ACE data sets and compares favorably to a modified version of their model. Despite these improvements, its performance is still not comparable to that of a fully supervised coreference resolver.

A natural way to extend these unsupervised coreference models is to incorporate additional linguistic knowledge sources, such as those employed by our fully supervised resolver. However, feature engineering is in general more difficult for generative models than for discriminative models, as the former typically require non-overlapping features. We plan to explore this possibility in future work.

Acknowledgments

We thank the three anonymous reviewers for their comments on an earlier draft of the paper. This work was supported in part by NSF Grant 0812261.

References

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of COLT*, pages 92–100.

Colin Cherry and Shane Bergsma. 2005. An expectation maximization approach to pronoun resolution. In *Proceedings of CoNLL*, pages 88–95.

Hal Daumé III and Daniel Marcu. 2005. A large-scale exploration of effective global features for a joint entity detection and tracking model. In *Proceedings of HLT/EMNLP*, pages 97–104.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Pascal Denis and Jason Baldridge. 2007. Global, joint determination of anaphoricity and coreference resolution using integer programming. In *Proceedings of NAACL/HLT*, pages 236–243.

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.

Aria Haghighi and Dan Klein. 2007. Unsupervised coreference resolution in a nonparametric Bayesian model. In *Proceedings of the ACL*, pages 848–855.

Heng Ji, David Westbrook, and Ralph Grishman. 2005. Using semantic relations to refine coreference decisions. In *Proceedings of HLT/EMNLP*, pages 17–24.

Andrew Kehler, Douglas Appelt, Lara Taylor, and Aleksandr Simma. 2004. Competitive self-trained pronoun interpretation In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 33–36.

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*, pages 135–142.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of HLT/EMNLP*, pages 25–32.

Christoph Müller, Stefan Rapp, and Michael Strube. 2002. Applying co-training to reference resolution. In *Proceedings of the ACL*, pages 352–359.

Vincent Ng. 2007. Shallow semantics for coreference resolution. In *Proceedings of IJCAI*, pages 1689–1694.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the ACL*, pages 104–111.

Vincent Ng and Claire Cardie. 2003. Weakly supervised natural language learning without redundant views. In *HLT-NAACL: Main Proceedings*, pages 173–180.

Simone Paolo Ponzetto and Michael Strube. 2006. Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. In *Proceedings of HLT/NAACL*, pages 192–199.

J. Ross Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Michael Strube, Stefan Rapp, and Christoph Müller. 2002. The influence of minimum edit distance on reference resolution. In *Proceedings of EMNLP*, pages 312–319.

Yee Whye Teh, Michael Jordan, Matthew Beal, and David Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1527–1554.

Renata Vieira and Massimo Poesio. 2000. An empirically-based system for processing definite descriptions. *Computational Linguistics*, 26(4):539–593.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of MUC-6*, pages 45–52.

Xiaofeng Yang, GuoDong Zhou, Jian Su, and Chew Lim Tan. 2003. Coreference resolution using competitive learning approach. In *Proceedings of the ACL*, pages 176–183.

Joint Unsupervised Coreference Resolution with Markov Logic

Hoifung Poon **Pedro Domingos**
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350, U.S.A.
{hoifung,pedrod}@cs.washington.edu

Abstract

Machine learning approaches to coreference resolution are typically supervised, and require expensive labeled data. Some unsupervised approaches have been proposed (e.g., Haghighi and Klein (2007)), but they are less accurate. In this paper, we present the first unsupervised approach that is competitive with supervised ones. This is made possible by performing joint inference across mentions, in contrast to the pairwise classification typically used in supervised methods, and by using Markov logic as a representation language, which enables us to easily express relations like apposition and predicate nominals. On MUC and ACE datasets, our model outperforms Haghigi and Klein's one using only a fraction of the training data, and often matches or exceeds the accuracy of state-of-the-art supervised models.

1 Introduction

The goal of coreference resolution is to identify *mentions* (typically noun phrases) that refer to the same *entities*. This is a key subtask in many NLP applications, including information extraction, question answering, machine translation, and others. Supervised learning approaches treat the problem as one of classification: for each pair of mentions, predict whether they corefer or not (e.g., McCallum & Wellner (2005)). While successful, these approaches require labeled training data, consisting of mention pairs and the correct decisions for them. This limits their applicability.

Unsupervised approaches are attractive due to the availability of large quantities of unlabeled text. However, unsupervised coreference resolution is much more difficult. Haghighi and Klein's (2007) model, the most sophisticated to date, still lags supervised ones by a substantial margin. Extending it appears difficult, due to the limitations of its Dirichlet process-based representation.

The lack of label information in unsupervised coreference resolution can potentially be overcome by performing joint inference, which leverages the "easy" decisions to help make related "hard" ones. Relations that have been exploited in supervised coreference resolution include transitivity (McCallum & Wellner, 2005) and anaphoricity (Denis & Baldridge, 2007). However, there is little work to date on joint inference for unsupervised resolution.

We address this problem using Markov logic, a powerful and flexible language that combines probabilistic graphical models and first-order logic (Richardson & Domingos, 2006). Markov logic allows us to easily build models involving relations among mentions, like apposition and predicate nominals. By extending the state-of-the-art algorithms for inference and learning, we developed the first general-purpose unsupervised learning algorithm for Markov logic, and applied it to unsupervised coreference resolution.

We test our approach on standard MUC and ACE datasets. Our basic model, trained on a minimum of data, suffices to outperform Haghighi and Klein's (2007) one. Our full model, using apposition and other relations for joint inference, is often as accurate as the best supervised models, or more.

We begin by reviewing the necessary background on Markov logic. We then describe our Markov logic network for joint unsupervised coreference resolution, and the learning and inference algorithms we used. Finally, we present our experiments and results.

2 Related Work

Most existing supervised learning approaches for coreference resolution are suboptimal since they resolve each mention pair independently, only imposing transitivity in postprocessing (Ng, 2005). Moreover, many of them break up the resolution step into subtasks (e.g., first determine whether a mention is anaphoric, then classify whether it is coreferent with an antecedent), which further forsakes opportunities for joint inference that have been shown to be helpful (Poon & Domingos, 2007). Using graph partitioning, McCallum & Wellner (2005) incorporated transitivity into pairwise classification and achieved the state-of-the-art result on the MUC-6 dataset, but their approach can only leverage one binary relation at a time, not arbitrary relations among mentions. Denis & Baldridge (2007) determined anaphoricity and pairwise classification jointly using integer programming, but they did not incorporate transitivity or other relations.

While potentially more appealing, unsupervised learning is very challenging, and unsupervised coreference resolution systems are still rare to this date. Prior to our work, the best performance in unsupervised coreference resolution was achieved by Haghighi & Klein (2007), using a nonparametric Bayesian model based on hierarchical Dirichlet processes. At the heart of their system is a mixture model with a few linguistically motivated features such as head words, entity properties and salience. Their approach is a major step forward in unsupervised coreference resolution, but extending it is challenging. The main advantage of Dirichlet processes is that they are exchangeable, allowing parameters to be integrated out, but Haghighi and Klein forgo this when they introduce salience. Their model thus requires Gibbs sampling over both assignments and parameters, which can be very expensive. Haghighi and Klein circumvent this by making approximations that potentially hurt accuracy. At the same time, the Dirichlet process prior favors skewed cluster sizes and a number of clusters that grows logarithmically with the number of data points, neither of which seems generally appropriate for coreference resolution.

Further, deterministic or strong non-deterministic dependencies cause Gibbs sampling to break down (Poon & Domingos, 2006), making it difficult to leverage many linguistic regularities. For example, apposition (as in "Bill Gates, the chairman of Microsoft") suggests coreference, and thus the two mentions it relates should always be placed in the same cluster. However, Gibbs sampling can only move one mention at a time from one cluster to another, and this is unlikely to happen, because it would require breaking the apposition rule. Blocked sampling can alleviate this problem by sampling multiple mentions together, but it requires that the block size be predetermined to a small fixed number. When we incorporate apposition and other regularities the blocks can become arbitrarily large, making this infeasible. For example, suppose we also want to leverage predicate nominals (i.e., the subject and the predicating noun of a copular verb are likely coreferent). Then a sentence like "He is Bill Gates, the chairman of Microsoft" requires a block of four mentions: "He", "Bill Gates", "the chairman of Microsoft", and "Bill Gates, the chairman of Microsoft". Similar difficulties occur with other inference methods. Thus, extending Haghighi and Klein's model to include richer linguistic features is a challenging problem.

Our approach is instead based on Markov logic, a powerful representation for joint inference with uncertainty (Richardson & Domingos, 2006). Like Haghighi and Klein's, our model is cluster-based rather than pairwise, and implicitly imposes transitivity. We do not predetermine anaphoricity of a mention, but rather fuse it into the integrated resolution process. As a result, our model is inherently joint among mentions and subtasks. It shares several features with Haghighi & Klein's model, but removes or refines features where we believe it is appropriate to. Most importantly, our model leverages apposition and predicate nominals, which Haghighi & Klein did not use. We show that this can be done very easily in our framework, and yet results in very substantial accuracy gains.

It is worth noticing that Markov logic is also well suited for joint inference in supervised systems (e.g., transitivity, which took McCallum & Wellner (2005) nontrivial effort to incorporate, can be handled in Markov logic with the addition of a single formula (Poon & Domingos, 2008)).

3 Markov Logic

In many NLP applications, there exist rich relations among objects, and recent work in statistical relational learning (Getoor & Taskar, 2007) and structured prediction (Bakir *et al.*, 2007) has shown that leveraging these can greatly improve accuracy. One of the most powerful representations for joint inference is Markov logic, a probabilistic extension of first-order logic (Richardson & Domingos, 2006). A *Markov logic network (MLN)* is a set of weighted first-order clauses. Together with a set of constants, it defines a Markov network with one node per ground atom and one feature per ground clause. The weight of a feature is the weight of the first-order clause that originated it. The probability of a state x in such a network is given by $P(x) = (1/Z)\exp\left(\sum_i w_i f_i(x)\right)$, where Z is a normalization constant, w_i is the weight of the ith clause, $f_i = 1$ if the ith clause is true, and $f_i = 0$ otherwise.

Markov logic makes it possible to compactly specify probability distributions over complex relational domains. Efficient inference can be performed using MC-SAT (Poon & Domingos, 2006). MC-SAT is a "slice sampling" Markov chain Monte Carlo algorithm. Slice sampling introduces auxiliary variables u that decouple the original ones x, and alternately samples u conditioned on x and vice-versa. To sample from the slice (the set of states x consistent with the current u), MC-SAT calls SampleSAT (Wei *et al.*, 2004), which uses a combination of satisfiability testing and simulated annealing. The advantage of using a satisfiability solver (WalkSAT) is that it efficiently finds isolated modes in the distribution, and as a result the Markov chain mixes very rapidly. The slice sampling scheme ensures that detailed balance is (approximately) preserved. MC-SAT is orders of magnitude faster than previous MCMC algorithms like Gibbs sampling, making efficient sampling possible on a scale that was previously out of reach.

Algorithm 1 MC-SAT(*clauses, weights, num_samples*)

```
$x^{(0)} \leftarrow$ Satisfy(hard *clauses*)
**for** $i \leftarrow 1$ to *num_samples* **do**
    $M \leftarrow \emptyset$
    **for all** $c_k \in$ *clauses* satisfied by $x^{(i-1)}$ **do**
        With probability $1 - e^{-w_k}$ add $c_k$ to $M$
    **end for**
    Sample $x^{(i)} \sim \mathcal{U}_{SAT(M)}$
**end for**
```

Algorithm 1 gives pseudo-code for MC-SAT. At iteration $i-1$, the factor ϕ_k for clause c_k is either e^{w_k} if c_k is satisfied in $x^{(i-1)}$, or 1 otherwise. MC-SAT first samples the auxiliary variable u_k uniformly from $(0, \phi_k)$, then samples a new state uniformly from the set of states that satisfy $\phi'_k \geq u_k$ for all k (the slice). Equivalently, for each k, with probability $1 - e^{-w_k}$ the next state must satisfy c_k. In general, we can factorize the probability distribution in any way that facilitates inference, sample the u_k's, and make sure that the next state is drawn uniformly from solutions that satisfy $\phi'_k \geq u_k$ for all factors.

MC-SAT, like most existing relational inference algorithms, grounds all predicates and clauses, thus requiring memory and time exponential in the predicate and clause arities. We developed a general method for producing a "lazy" version of relational inference algorithms (Poon & Domingos, 2008), which carries exactly the same inference steps as the original algorithm, but only maintains a small subset of "active" predicates/clauses, grounding more as needed. We showed that Lazy-MC-SAT, the lazy version of MC-SAT, reduced memory and time by orders of magnitude in several domains. We use Lazy-MC-SAT in this paper.

Supervised learning for Markov logic maximizes the conditional log-likelihood $L(x, y) = \log P(Y = y|X = x)$, where Y represents the non-evidence predicates, X the evidence predicates, and x, y their values in the training data. For simplicity, from now on we omit X, whose values are fixed and always conditioned on. The optimization problem is convex and a global optimum can be found using gradient descent, with the gradient being

$$\begin{aligned}\frac{\partial}{\partial w_i}L(y) &= n_i(y) - \textstyle\sum_{y'} P(Y=y')n_i(y') \\ &= n_i(y) - E_Y[n_i].\end{aligned}$$

where n_i is the number of true groundings of clause i. The expected count can be approximated as

$$E_Y[n_i] \approx \frac{1}{N}\sum_{k=1}^{N} n_i(y_k)$$

where y_k are samples generated by MC-SAT. To combat overfitting, a Gaussian prior is imposed on all weights.

In practice, it is difficult to tune the learning rate for gradient descent, especially when the number of groundings varies widely among clauses. Lowd & Domingos (2007) used a preconditioned scaled conjugate gradient algorithm (PSCG) to address this problem. This estimates the optimal step size in each step as

$$\alpha = \frac{-d^T g}{d^T H d + \lambda d^T d}.$$

where g is the gradient, d the conjugate update direction, and λ a parameter that is automatically tuned to trade off second-order information with gradient descent. H is the Hessian matrix, with the (i,j)th entry being

$$\begin{aligned}\frac{\partial^2}{\partial w_i \partial w_j}L(y) &= E_Y[n_i]\cdot E_Y[n_j] - E_Y[n_i \cdot n_j] \\ &= -Cov_Y[n_i, n_j].\end{aligned}$$

The Hessian can be approximated with the same samples used for the gradient. Its negative inverse diagonal is used as the preconditioner.[1]

The open-source Alchemy package (Kok *et al.*, 2007) provides implementations of existing algorithms for Markov logic. In Section 5, we develop the first general-purpose unsupervised learning algorithm for Markov logic by extending the existing algorithms to handle hidden predicates.[2]

[1]Lowd & Domingos showed that α can be computed more efficiently, without explicitly approximating or storing the Hessian. Readers are referred to their paper for details.

[2]Alchemy includes a discriminative EM algorithm, but it assumes that only a few values are missing, and cannot handle completely hidden predicates. Kok & Domingos (2007) applied Markov logic to relational clustering, but they used hard EM.

4 An MLN for Joint Unsupervised Coreference Resolution

In this section, we present our MLN for joint unsupervised coreference resolution. Our model deviates from Haghighi & Klein's (2007) in several important ways. First, our MLN does not model saliences for proper nouns or nominals, as their influence is marginal compared to other features; for pronoun salience, it uses a more intuitive and simpler definition based on distance, and incorporated it as a prior. Another difference is in identifying heads. For the ACE datasets, Haghighi and Klein used the gold heads; for the MUC-6 dataset, where labels are not available, they crudely picked the rightmost token in a mention. We show that a better way is to determine the heads using head rules in a parser. This improves resolution accuracy and is always applicable. Crucially, our MLN leverages syntactic relations such as apposition and predicate nominals, which are not used by Haghighi and Klein. In our approach, what it takes is just adding two formulas to the MLN.

As common in previous work, we assume that true mention boundaries are given. We do not assume any other labeled information. In particular, we do not assume gold name entity recognition (NER) labels, and unlike Haghighi & Klein (2007), we do not assume gold mention types (for ACE datasets, they also used gold head words). We determined the head of a mention either by taking its rightmost token, or by using the head rules in a parser. We detected pronouns using a list.

4.1 Base MLN

The main query predicate is $\mathtt{InClust(m,c!)}$, which is true iff mention $\mathtt{m}$ is in cluster $\mathtt{c}$. The "$\mathtt{t!}$" notation signifies that for each $\mathtt{m}$, this predicate is true for a unique value of $\mathtt{c}$. The main evidence predicate is $\mathtt{Head(m,t!)}$, where $\mathtt{m}$ is a mention and $\mathtt{t}$ a token, and which is true iff $\mathtt{t}$ is the head of $\mathtt{m}$. A key component in our MLN is a simple head mixture model, where the mixture component priors are represented by the unit clause

$$\mathtt{InClust(+m,+c)}$$

and the head distribution is represented by the *head prediction rule*

$$\mathtt{InClust(m,+c)} \wedge \mathtt{Head(m,+t)}.$$

All free variables are implicitly universally quantified. The "+" notation signifies that the MLN contains an instance of the rule, with a separate weight, for each value combination of the variables with a plus sign.

By convention, at each inference step we name each non-empty cluster after the earliest mention it contains. This helps break the symmetry among mentions, which otherwise produces multiple optima and makes learning unnecessarily harder. To encourage clustering, we impose an exponential prior on the number of non-empty clusters with weight -1.

The above model only clusters mentions with the same head, and does not work well for pronouns. To address this, we introduce the predicate $\texttt{IsPrn}(\texttt{m})$, which is true iff the mention $\texttt{m}$ is a pronoun, and adapt the head prediction rule as follows:

$$\neg\texttt{IsPrn}(\texttt{m}) \wedge \texttt{InClust}(\texttt{m}, +\texttt{c}) \wedge \texttt{Head}(\texttt{m}, +\texttt{t})$$

This is always false when $\texttt{m}$ is a pronoun, and thus applies only to non-pronouns.

Pronouns tend to resolve with mentions that are semantically compatible with them. Thus we introduce predicates that represent entity type, number, and gender: $\texttt{Type}(\texttt{x}, \texttt{e}!)$, $\texttt{Number}(\texttt{x}, \texttt{n}!)$, $\texttt{Gender}(\texttt{x}, \texttt{g}!)$, where $\texttt{x}$ can be either a cluster or mention, $\texttt{e} \in \{\texttt{Person}, \texttt{Organization}, \texttt{Location}, \texttt{Other}\}$, $\texttt{n} \in \{\texttt{Singular}, \texttt{Plural}\}$ and $\texttt{g} \in \{\texttt{Male}, \texttt{Female}, \texttt{Neuter}\}$. Many of these are known for pronouns, and some can be inferred from simple linguistic cues (e.g., "Ms. Galen" is a singular female person, while "XYZ Corp." is an organization).[3] Entity type assignment is represented by the unit clause

$$\texttt{Type}(+\texttt{x}, +\texttt{e})$$

and similarly for number and gender. A mention should agree with its cluster in entity type. This is ensured by the hard rule (which has infinite weight and must be satisfied)

$$\texttt{InClust}(\texttt{m}, \texttt{c}) \Rightarrow (\texttt{Type}(\texttt{m}, \texttt{e}) \Leftrightarrow \texttt{Type}(\texttt{c}, \texttt{e}))$$

[3]We used the following cues: Mr., Ms., Jr., Inc., Corp., corporation, company. The proportions of known properties range from 14% to 26%.

There are similar hard rules for number and gender.

Different pronouns prefer different entity types, as represented by

$$\texttt{IsPrn}(\texttt{m}) \wedge \texttt{InClust}(\texttt{m}, \texttt{c}) \\ \wedge\texttt{Head}(\texttt{m}, +\texttt{t}) \wedge \texttt{Type}(\texttt{c}, +\texttt{e})$$

which only applies to pronouns, and whose weight is positive if pronoun $\texttt{t}$ is likely to assume entity type $\texttt{e}$ and negative otherwise. There are similar rules for number and gender.

Aside from semantic compatibility, pronouns tend to resolve with nearby mentions. To model this, we impose an exponential prior on the distance (number of mentions) between a pronoun and its antecedent, with weight -1.[4] This is similar to Haghighi and Klein's treatment of salience, but simpler.

4.2 Full MLN

Syntactic relations among mentions often suggest coreference. Incorporating such relations into our MLN is straightforward. We illustrate this with two examples: apposition and predicate nominals. We introduce a predicate for apposition, $\texttt{Appo}(\texttt{x}, \texttt{y})$, where $\texttt{x}, \texttt{y}$ are mentions, and which is true iff $\texttt{y}$ is an appositive of $\texttt{x}$. We then add the rule

$$\texttt{Appo}(\texttt{x}, \texttt{y}) \Rightarrow (\texttt{InClust}(\texttt{x}, \texttt{c}) \Leftrightarrow \texttt{InClust}(\texttt{y}, \texttt{c}))$$

which ensures that x, y are in the same cluster if y is an appositive of x. Similarly, we introduce a predicate for predicate nominals, $\texttt{PredNom}(\texttt{x}, \texttt{y})$, and the corresponding rule.[5] The weights of both rules can be learned from data with a positive prior mean. For simplicity, in this paper we treat them as hard constraints.

4.3 Rule-Based MLN

We also consider a rule-based system that clusters non-pronouns by their heads, and attaches a pronoun to the cluster which has no known conflicting

[4]For simplicity, if a pronoun has no antecedent, we define the distance to be ∞. So a pronoun must have an antecedent in our model, unless it is the first mention in the document or it can not resolve with previous mentions without violating hard constraints. It is straightforward to soften this with a finite penalty.

[5]We detected apposition and predicate nominatives using simple heuristics based on parses, e.g., if (NP, comma, NP) are the first three children of an NP, then any two of the three noun phrases are apposition.

type, number, or gender, and contains the closest antecedent for the pronoun. This system can be encoded in an MLN with just four rules. Three of them are the ones for enforcing agreement in type, number, and gender between a cluster and its members, as defined in the base MLN. The fourth rule is

$$\begin{gathered}\neg\texttt{IsPrn(m1)} \wedge \neg\texttt{IsPrn(m2)} \\ \wedge\texttt{Head(m1,h1)} \wedge \texttt{Head(m2,h2)} \\ \wedge\texttt{InClust(m1,c1)} \wedge \texttt{InClust(m2,c2)} \\ \Rightarrow (\texttt{c1} = \texttt{c2} \Leftrightarrow \texttt{h1} = \texttt{h2}).\end{gathered}$$

With a large but not infinite weight (e.g., 100), this rule has the effect of clustering non-pronouns by their heads, except when it violates the hard rules. The MLN can also include the apposition and predicate-nominal rules. As in the base MLN, we impose the same exponential prior on the number of non-empty clusters and that on the distance between a pronoun and its antecedent. This simple MLN is remarkably competitive, as we will see in the experiment section.

5 Learning and Inference

Unsupervised learning in Markov logic maximizes the conditional log-likelihood

$$\begin{aligned} L(x,y) &= \log P(Y=y|X=x) \\ &= \log \textstyle\sum_z P(Y=y, Z=z|X=x) \end{aligned}$$

where Z are unknown predicates. In our coreference resolution MLN, Y includes Head and known groundings of Type, Number and Gender, Z includes InClust and unknown groundings of Type, Number, Gender, and X includes IsPrn, Appo and PredNom. (For simplicity, from now on we drop X from the formula.) With Z, the optimization problem is no longer convex. However, we can still find a local optimum using gradient descent, with the gradient being

$$\frac{\partial}{\partial w_i} L(y) = E_{Z|y}[n_i] - E_{Y,Z}[n_i]$$

where n_i is the number of true groundings of the ith clause. We extended PSCG for unsupervised learning. The gradient is the difference of two expectations, each of which can be approximated using samples generated by MC-SAT. The (i, j)th entry of the Hessian is now

$$\frac{\partial^2}{\partial w_i \partial w_j} L(y) = Cov_{Z|y}[n_i, n_j] - Cov_{Y,Z}[n_i, n_j]$$

and the step size can be computed accordingly. Since our problem is no longer convex, the negative diagonal Hessian may contain zero or negative entries, so we first took the absolute values of the diagonal and added 1, then used the inverse as the preconditioner. We also adjusted λ more conservatively than Lowd & Domingos (2007).

Notice that when the objects form independent subsets (in our cases, mentions in each document), we can process them in parallel and then gather sufficient statistics for learning. We developed an efficient parallelized implementation of our unsupervised learning algorithm using the message-passing interface (MPI). Learning in MUC-6 took only one hour, and in ACE-2004 two and a half.

To reduce burn-in time, we initialized MC-SAT with the state returned by MaxWalkSAT (Kautz *et al.*, 1997), rather than a random solution to the hard clauses. In the existing implementation in Alchemy (Kok *et al.*, 2007), SampleSAT flips only one atom in each step, which is inefficient for predicates with unique-value constraints (e.g., Head(m, c!)). Such predicates can be viewed as multi-valued predicates (e.g., Head(m) with value ranging over all c's) and are prevalent in NLP applications. We adapted SampleSAT to flip two or more atoms in each step so that the unique-value constraints are automatically satisfied. By default, MC-SAT treats each ground clause as a separate factor while determining the slice. This can be very inefficient for highly correlated clauses. For example, given a non-pronoun mention m currently in cluster c and with head t, among the mixture prior rules involving m InClust(m, c) is the only one that is satisfied, and among those head-prediction rules involving m, $\neg$IsPrn(m) $\wedge$ InClust(m, c) $\wedge$ Head(m, t) is the only one that is satisfied; the factors for these rules multiply to $\phi = \exp(w_{\texttt{m,c}} + w_{\texttt{m,c,t}})$, where $w_{\texttt{m,c}}$ is the weight for InClust(m, c), and $w_{\texttt{m,c,t}}$ is the weight for $\neg$IsPrn(m) $\wedge$ InClust(m, c) $\wedge$ Head(m, t), since an unsatisfied rule contributes a factor of $e^0 = 1$. We extended MC-SAT to treat each set of mutually exclusive and exhaustive rules as a single factor. E.g., for the above m, MC-SAT now samples u uniformly from $(0, \phi)$, and requires that in the next state ϕ' be no less than u. Equivalently, the new cluster and head for $\mathtt{m}$ should satisfy $w_{\mathtt{m},\mathtt{c}'} + w_{\mathtt{m},\mathtt{c}',\mathtt{t}'} \geq \log(u)$. We extended SampleSAT so that when it considers flipping any variable involved in such constraints (e.g., `c` or `t` above), it ensures that their new values still satisfy these constraints.

The final clustering is found using the MaxWalkSAT weighted satisfiability solver (Kautz *et al.*, 1997), with the appropriate extensions. We first ran a MaxWalkSAT pass with only finite-weight formulas, then ran another pass with all formulas. We found that this significantly improved the quality of the results that MaxWalkSAT returned.

6 Experiments

6.1 System

We implemented our method as an extension to the Alchemy system (Kok *et al.*, 2007). Since our learning uses sampling, all results are the average of five runs using different random seeds. Our optimization problem is not convex, so initialization is important. The core of our model (head mixture) tends to cluster non-pronouns with the same head. Therefore, we initialized by setting all weights to zero, and running the same learning algorithm on the base MLN, while assuming that in the ground truth, non-pronouns are clustered by their heads. (Effectively, the corresponding `InClust` atoms are assigned to appropriate values and are included in Y rather than Z during learning.) We used 30 iterations of PSCG for learning. (In preliminary experiments, additional iterations had little effect on coreference accuracy.) We generated 100 samples using MC-SAT for each expectation approximation.[6]

6.2 Methodology

We conducted experiments on MUC-6, ACE-2004, and ACE Phrase-2 (ACE-2). We evaluated our systems using two commonly-used scoring programs: MUC (Vilain *et al.*, 1995) and B^3 (Amit & Baldwin, 1998). To gain more insight, we also report pairwise resolution scores and mean absolute error in the number of clusters.

[6]Each sample actually contains a large number of groundings, so 100 samples yield sufficiently accurate statistics for learning.

The MUC-6 dataset consists of 30 documents for testing and 221 for training. To evaluate the contribution of the major components in our model, we conducted five experiments, each differing from the previous one in a single aspect. We emphasize that our approach is unsupervised, and thus the data only contains raw text plus true mention boundaries.

MLN-1 In this experiment, the base MLN was used, and the head was chosen crudely as the rightmost token in a mention. Our system was run on each test document separately, using a minimum of training data (the document itself).

MLN-30 Our system was trained on all 30 test documents together. This tests how much can be gained by pooling information.

MLN-H The heads were determined using the head rules in the Stanford parser (Klein & Manning, 2003), plus simple heuristics to handle suffixes such as "Corp." and "Inc."

MLN-HA The apposition rule was added.

MLN-HAN The predicate-nominal rule was added. This is our full model.

We also compared with two rule-based MLNs: **RULE** chose the head crudely as the rightmost token in a mention, and did not include the apposition rule and predicate-nominal rule; **RULE-HAN** chose the head using the head rules in the Stanford parser, and included the apposition rule and predicate-nominal rule.

Past results on ACE were obtained on different releases of the datasets, e.g., Haghighi and Klein (2007) used the ACE-2004 training corpus, Ng (2005) and Denis and Baldridge (2007) used ACE Phrase-2, and Culotta *et al.* (2007) used the ACE-2004 formal test set. In this paper, we used the ACE-2004 training corpus and ACE Phrase-2 (ACE-2) to enable direct comparisons with Haghighi & Klein (2007), Ng (2005), and Denis and Baldridge (2007). Due to license restrictions, we were not able to obtain the ACE-2004 formal test set and so cannot compare directly to Culotta *et al.* (2007). The English version of the ACE-2004 training corpus contains two sections, BNEWS and NWIRE, with 220 and 128 documents, respectively. ACE-2 contains a

Table 1: Comparison of coreference results in MUC scores on the MUC-6 dataset.

	# Doc.	Prec.	Rec.	F1
H&K	60	80.8	52.8	63.9
H&K	381	80.4	62.4	70.3
M&W	221	-	-	73.4
RULE	-	76.0	65.9	70.5
RULE-HAN	-	81.3	72.7	76.7
MLN-1	1	76.5	66.4	71.1
MLN-30	30	77.5	67.3	72.0
MLN-H	30	81.8	70.1	75.5
MLN-HA	30	82.7	75.1	78.7
MLN-HAN	30	83.0	75.8	79.2

Table 2: Comparison of coreference results in MUC scores on the ACE-2004 (English) datasets.

EN-BNEWS	Prec.	Rec.	F1
H&K	63.2	61.3	62.3
MLN-HAN	66.8	67.8	67.3
EN-NWIRE	Prec.	Rec.	F1
H&K	66.7	62.3	64.2
MLN-HAN	71.3	70.5	70.9

training set and a test set. In our experiments, we only used the test set, which contains three sections, BNEWS, NWIRE, and NPAPER, with 51, 29, and 17 documents, respectively.

6.3 Results

Table 1 compares our system with previous approaches on the MUC-6 dataset, in MUC scores. Our approach greatly outperformed Haghighi & Klein (2007), the state-of-the-art unsupervised system. Our system, trained on individual documents, achieved an F1 score more than 7% higher than theirs trained on 60 documents, and still outperformed it trained on 381 documents. Training on the 30 test documents together resulted in a significant gain. (We also ran experiments using more documents, and the results were similar.) Better head identification (MLN-H) led to a large improvement in accuracy, which is expected since for mentions with a right modifier, the rightmost tokens confuse rather than help coreference (e.g., "the chairman of Microsoft"). Notice that with this improvement our system already outperforms a state-of-the-art supervised system (McCallum & Wellner, 2005). Leveraging apposition resulted in another large improvement, and predicate nominals also helped. Our full model scores about 9% higher than Haghighi & Klein (2007), and about 6% higher than McCallum & Wellner (2005). To our knowledge, this is the best coreference accuracy reported on MUC-6 to date.[7] The B^3 scores of MLN-HAN on the MUC-6 dataset are 77.4 (precision), 67.6 (recall) and 72.2 (F1). (The other systems did not report B^3.) Interestingly, the rule-based MLN (**RULE**) sufficed to outperform Haghighi & Klein (2007), and by using better heads and the apposition and predicate-nominal rules (**RULE-HAN**), it outperformed McCallum & Wellner (2005), the supervised system. The MLNs with learning (**MLN-30** and **MLN-HAN**), on the

Table 3: Comparison of coreference results in MUC scores on the ACE-2 datasets.

BNEWS	Prec.	Rec.	F1
Ng	67.9	62.2	64.9
D&B	78.0	62.1	69.2
MLN-HAN	68.3	66.6	67.4
NWIRE	Prec.	Rec.	F1
Ng	60.3	50.1	54.7
D&B	75.8	60.8	67.5
MLN-HAN	67.7	67.3	67.4
NPAPER	Prec.	Rec.	F1
Ng	71.4	67.4	69.3
D&B	77.6	68.0	72.5
MLN-HAN	69.2	71.7	70.4

Table 4: Comparison of coreference results in B^3 scores on the ACE-2 datasets.

BNEWS	Prec.	Rec.	F1
Ng	77.1	57.0	65.6
MLN-HAN	70.3	65.3	67.7
NWIRE	Prec.	Rec.	F1
Ng	75.4	59.3	66.4
MLN-HAN	74.7	68.8	71.6
NPAPER	Prec.	Rec.	F1
Ng	75.4	59.3	66.4
MLN-HAN	70.0	66.5	68.2

[7]As pointed out by Haghighi & Klein (2007), Luo *et al.* (2004) obtained a very high accuracy on MUC-6, but their system used gold NER features and is not directly comparable.

Table 5: Our coreference results in precision, recall, and F1 for pairwise resolution.

Pairwise	Prec.	Rec.	F1
MUC-6	63.0	57.0	59.9
EN-BNEWS	51.2	36.4	42.5
EN-NWIRE	62.6	38.9	48.0
BNEWS	44.6	32.3	37.5
NWIRE	59.7	42.1	49.4
NPAPER	64.3	43.6	52.0

Table 6: Average gold number of clusters per document vs. the mean absolute error of our system.

# Clusters	MUC-6	EN-BN	EN-NW
Gold	15.4	22.3	37.2
Mean Error	4.7	3.0	4.8
# Clusters	BNEWS	NWIRE	NPAPER
Gold	20.4	39.2	55.2
Mean Error	2.5	5.6	6.6

other hand, substantially outperformed the corresponding rule-based ones.

Table 2 compares our system to Haghighi & Klein (2007) on the ACE-2004 training set in MUC scores. Again, our system outperformed theirs by a large margin. The B^3 scores of MLN-HAN on the ACE-2004 dataset are 71.6 (precision), 68.4 (recall) and 70.0 (F1) for BNEWS, and 75.7 (precision), 69.2 (recall) and 72.3 (F1) for NWIRE. (Haghighi & Klein (2007) did not report B^3.) Due to license restrictions, we could not compare directly to Culotta *et al.* (2007), who reported overall B^3-F1 of 79.3 on the formal test set.

Tables 3 and 4 compare our system to two recent supervised systems, Ng (2005) and Denis & Baldridge (2007). Our approach significantly outperformed Ng (2005). It tied with Denis & Baldridge (2007) on NWIRE, and was somewhat less accurate on BNEWS and NPAPER.

Luo *et al.* (2004) pointed out that one can obtain a very high MUC score simply by lumping all mentions together. B^3 suffers less from this problem but is not perfect. Thus we also report pairwise resolution scores (Table 5), the gold number of clusters, and our mean absolute error in the number of clusters (Table 6). Systems that simply merge all mentions will have exceedingly low pairwise precision (far below 50%), and very large errors in the number of clusters. Our system has fairly good pairwise precisions and small mean error in the number of clusters, which verifies that our results are sound.

6.4 Error Analysis

Many of our system's remaining errors involve nominals. Additional features should be considered to distinguish mentions that have the same head but are different entities. For pronouns, many remaining errors can be corrected using linguistic knowledge like binding theory and salience hierarchy. Our heuristics for identifying appositives and predicate nominals also make many errors, which often can be fixed with additional name entity recognition capabilities (e.g., given "Mike Sullivan, VOA News", it helps to know that the former is a person and the latter an organization). The most challenging case involves phrases with different heads that are both proper nouns (e.g., "Mr. Bush" and "the White House"). Handling these cases requires domain knowledge and/or more powerful joint inference.

7 Conclusion

This paper introduces the first unsupervised coreference resolution system that is as accurate as supervised systems. It performs joint inference among mentions, using relations like apposition and predicate nominals. It uses Markov logic as a representation language, which allows it to be easily extended to incorporate additional linguistic and world knowledge. Future directions include incorporating additional knowledge, conducting joint entity detection and coreference resolution, and combining coreference resolution with other NLP tasks.

8 Acknowledgements

We thank the anonymous reviewers for their comments. This research was funded by DARPA contracts NBCH-D030010/02-000225, FA8750-07-D-0185, and HR0011-07-C-0060, DARPA grant FA8750-05-2-0283, NSF grant IIS-0534881, and ONR grant N-00014-05-1-0313 and N00014-08-1-0670. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, NSF, ONR, or the United States Government.

References

Amit, B. & Baldwin, B. 1998. Algorithms for scoring coreference chains. In *Proc. MUC-7*.

Bakir, G.; Hofmann, T.; Schölkopf, B.; Smola, A.; Taskar, B. and Vishwanathan, S. (eds.) 2007. *Predicting Structured Data*. MIT Press.

Culotta, A.; Wick, M.; Hall, R. and McCallum, A. 2007. First-order probabilistic models for coreference resolution. In *Proc. NAACL-07*.

Denis, P. & Baldridge, J. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. In *Proc. NAACL-07*.

Getoor, L. & Taskar, B. (eds.) 2007. *Introduction to Statistical Relational Learning*. MIT Press.

Haghighi, A. & Klein, D. 2007. Unsupervised coreference resolution in a nonparametric Bayesian model. In *Proc. ACL-07*.

Kautz, H.; Selman, B.; and Jiang, Y. 1997. A general stochastic approach to solving problems with hard and soft constraints. In *The Satisfiability Problem: Theory and Applications*. AMS.

Klein, D. & Manning, C. 2003. Accurate unlexicalized parsing. In *Proc. ACL-03*.

Kok, S.; Singla, P.; Richardson, M.; Domingos, P.; Sumner, M.; Poon, H. & Lowd, D. 2007. The Alchemy system for statistical relational AI. http://alchemy.cs.washington.edu/.

Lowd, D. & Domingos, D. 2007. Efficient weight learning for Markov logic networks. In *Proc. PKDD-07*.

Luo, X.; Ittycheriah, A.; Jing, H.; Kambhatla, N. and Roukos, S. 2004. A mention-synchronous coreference resolution algorithm based on the bell tree. In *Proc. ACL-04*.

McCallum, A. & Wellner, B. 2005. Conditional models of identity uncertainty with application to noun coreference. In *Proc. NIPS-04*.

Ng, V. 2005. Machine Learning for Coreference Resolution: From Local Classification to Global Ranking. In *Proc. ACL-05*.

Poon, H. & Domingos, P. 2006. Sound and efficient inference with probabilistic and deterministic dependencies. In *Proc. AAAI-06*.

Poon, H. & Domingos, P. 2007. Joint inference in information extraction. In *Proc. AAAI-07*.

Poon, H. & Domingos, P. 2008. A general method for reducing the complexity of relational inference and its application to MCMC. In *Proc. AAAI-08*.

Richardson, M. & Domingos, P. 2006. Markov logic networks. *Machine Learning* 62:107–136.

Vilain, M.; Burger, J.; Aberdeen, J.; Connolly, D. & Hirschman, L. 1995. A model-theoretic coreference scoring scheme. In *Proc. MUC-6*.

Wei, W.; Erenrich, J. and Selman, B. 2004. Towards efficient sampling: Exploiting random walk strategies. In *Proc. AAAI-04*.

Specialized models and ranking for coreference resolution

Pascal Denis
ALPAGE Project Team
INRIA Rocquencourt
F-78153 Le Chesnay, France
pascal.denis@inria.fr

Jason Baldridge
Department of Linguistics
University of Texas at Austin
Austin, TX 78712-0198, USA
jbaldrid@mail.utexas.edu

Abstract

This paper investigates two strategies for improving coreference resolution: (1) training separate models that specialize in particular types of mentions (e.g., pronouns versus proper nouns) and (2) using a ranking loss function rather than a classification function. In addition to being conceptually simple, these modifications of the standard single-model, classification-based approach also deliver significant performance improvements. Specifically, we show that on the ACE corpus both strategies produce f-score gains of more than 3% across the three coreference evaluation metrics (MUC, B^3, and CEAF).

1 Introduction

Coreference resolution is the task of partitioning a set of entity mentions in a text, where each partition corresponds to some entity in an underlying discourse model. While early machine learning approaches for the task relied on local, discriminative classifiers (Soon et al., 2001; Ng and Cardie, 2002b; Morton, 2000; Kehler et al., 2004), more recent approaches use joint and/or global models (McCallum and Wellner, 2004; Ng, 2004; Daumé III and Marcu, 2005; Denis and Baldridge, 2007a). This shift improves performance, but the systems are considerably more complex and often less efficient. Here, we explore two simple modifications of the first type of approach that yield performance gains which are comparable, and sometimes better, to those obtained with these more complex systems. These modifications involve: (i) the use of *rankers* instead of classifiers, and (ii) the use of linguistically motivated, *specialized models* for different types of mentions.

Ranking models provide a theoretically more adequate and empirically better alternative approach to pronoun resolution than standard classification-based approaches (Denis and Baldridge, 2007b). In essence, ranking models directly capture during training the competition among potential antecedent candidates, instead of considering them independently. This gives the ranker additional discriminative power and in turn better antecedent selection accuracy. Here, we show that ranking is also effective for the wider task of coreference resolution.

Coreference resolution involves several different types of anaphoric expressions: third-person pronouns, speech pronouns (i.e., first and second person pronouns), proper names, definite descriptions and other types of nominals (e.g., anaphoric uses of indefinite, quantified, and bare noun phrases). Different anaphoric expressions exhibit different patterns of resolution and are sensitive to different factors (Ariel, 1988; van der Sandt, 1992; Gundel et al., 1993), yet most machine learning approaches have ignored these differences and handle these different phenomena with a single, monolithic model. A few exceptions are worth noting. Morton (2000) and Ng (2005b) propose different classifiers models for different NPs for coreference resolution and pronoun resolution, respectively. Other partially capture the differential preferences between different anaphors via different sample selection strategies during training (Ng and Cardie, 2002b; Uryupina, 2004). More recently, Haghighi and Klein (2007) use the distinction between pronouns, nominals and proper nouns in their unsupervised, generative model for coreference resolution; for their model, this is absolutely critical for achieving better accuracy. Here, we show that using specialized models for different types of referential expressions improves performance for supervised models (both classifiers and rankers).

Both these strategies lead to improvements for all three standard coreference metrics: MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), and CEAF (Luo, 2005). In particular, our specialized ranker system provides absolute f-score improvements against an otherwise identical standard classifier system by 3.2%, 3.1%, and 3.6% for MUC, B^3, and CEAF, respectively.

2 Ranking

Numerous approaches to anaphora and coreference resolution reduce these tasks to a binary classification task, whereby *pairs* of mentions are classified as coreferential or not (McCarthy and Lehnert, 1995; Soon et al., 2001; Ng and Cardie, 2002b). Usually used in combination with a greedy right-to-left clustering, these approaches make very strong independence assumptions. Not only do they model each coreference decision separately, they actually model *each pair* of mentions as a separate event. Recasting these tasks as ranking tasks partly addresses this problem by directly making the comparison between different candidate antecedents for an anaphor part of the training criterion. Each candidate is assigned a conditional probability with respect to the *entire* candidate set. (Re)rankers have been successfully applied to numerous NLP tasks, such as parse selection (Osborne and Baldridge, 2004; Toutanova et al., 2004), parse reranking (Collins and Duffy, 2002; Charniak and Johnson, 2005), question-answering (Ravichandran et al., 2003).

The twin-candidate classification approach proposed by (Yang et al., 2003) shares some similarities with the ranker in making the comparison between candidate antecedents part of training. An important difference however is that under the twin-candidate approach, candidates are compared in *pairwise* fashion (and the best overall candidate is the one that has won the most round robin contests), while the ranker considers the entire candidate set at once. Another advantage of the ranking approach is that its complexity is only square in the number of mentions, while that of the twin-candidate model is cubic (see Denis and Baldridge (2007b) for a more detailed comparison in the context of pronoun resolution).

Our ranking models for coreference take the following log-linear form:

$$P_{rk}(\alpha_i|\pi) = \frac{\exp \sum_{j=1}^{m} w_j f_j(\pi, \alpha_i)}{\sum_{k} \exp \sum_{j=1}^{m} w_j f_j(\pi, \alpha_k)} \qquad (1)$$

where π stands for the anaphoric expression, α_i for an antecedent candidate, f_j the weighted features of the model. The denominator consists of a normalization factor over the k candidate mentions. Model parameters were estimated with the limited memory variable metric algorithm and Gaussian smoothing (σ^2=1000), using TADM (Malouf, 2002).

For the training of the different ranking models, we use the following procedure. For each model, instances are created by pairing each anaphor of the proper type (e.g., definite description) with a set of candidates which contains: (i) a true antecedent, and (ii) a set of non-antecedents. The selection of the true antecedent varies depending on the model we are training: for pronominal forms, the antecedent is selected as the *closest* preceding mention in the chain; for non-pronominal forms, we used the closest preceding *non-pronominal* mention in the chain as the antecedent. For the creation of the non-antecedent set, we collect all the non-antecedents that appear in a window of two sentences around the antecedent.[1] At test time, we consider *all* preceding mentions as potential antecedents.

Not all referential expressions in a given document are anaphors: some expressions introduce a discourse entity, rather than accessing an existing one. Thus, coreference resolvers must have a way of identifying such "discourse-new" expressions. This is easily handled in the standard classification approach: a mention will not be resolved if none of its candidates is classified positively (i.e., as coreferential). The problem is more troublesome for rankers, which always pick an antecedent from the candidate

[1]We suspect that different varying windows might be more appropriate for different types of expressions, but leaves this for further investigations.

set. A natural solution is to use a model that specifically predicts the discourse status (discourse-new vs. discourse-old) of each expression: only expressions that are classified as "discourse-old" by this model are considered by rankers.

Ng and Cardie (Ng and Cardie, 2002a) introduced the use of an "anaphoricity" classifier to act as a filter for coreference resolution in order to correct errors where antecedents are mistakenly identified for non-anaphoric mentions or antecedents are not determined for mentions which are indeed anaphoric. Their approach produced significant improvements in precision, but with consequent larger losses in recall. Ng (2004) improves recall by optimizing the anaphoricity threshold. By using joint inference for anaphoricity and coreference, Denis and Baldridge (2007a) avoid cascade-induced errors without the need to separately optimize the threshold.

We use a similar discourse status classifier to Ng and Cardie's as a filter on mentions for our rankers. We rely on three main types of information sources: (i) the form of mention (e.g., type of linguistic expression, number of tokens), (ii) positional features in the text, (iii) comparisons of the given mention to the mentions that precede it in the text. Evaluated on the ACE datasets, training the model on the `train` texts, and applying the classifier to the `devtest` texts, the model achieves an overall accuracy score of 80.8%, compared to a baseline of 59.7% when predicting the majority class ("discourse-old").

3 Specialized models

Our second strategy is to use different, specialized models for different referential expressions, similarly to Elwell and Baldridge's (2008) use of connective specific models for identifying the arguments of discourse connectives. For this, one must determine along which dimension to split such expressions. For example, Ng (2005b) learns models for each set of anaphors that are lexically identical (e.g., *I*, *he*, *they*, etc.). This option is possible for closed sets like pronouns, but not for other types of anaphors like proper names and definite descriptions. Another option is to rely on the particular *linguistic form* of the different expressions, as signaled by the head word category and the determiner (if any). More concretely, we use separate models for the following types: (i) third person pronouns, (ii) speech pronouns, (iii) proper names, (iv) definite descriptions, and (v) others (i.e., all expressions that don't fall into the previous categories).

The correlation between the form of a referential expression and its anaphoric behavior is actually central to various linguistic accounts (Prince, 1981; Ariel, 1988; Gundel et al., 1993). Basically, the idea is that linguistic form is an indicator of the status of the corresponding referent in the discourse model. That is, the use by the speaker of a particular linguistic form corresponds to a particular level of activation (or familiarity or salience or accessibility) in (what she thinks is) the addressee's discourse model. For many authors, the relation takes the form of a continuum and is often represented in the form of a referential hierarchy, such as:

> **Accessibility Hierarchy** (Ariel, 1988)
> Zero pronouns >> Pronouns >> Demonstrative pronouns >> Demonstrative NPs >> Short PNs >> Definite descriptions >> Full PNs >> Full PNs + appositive

The higher up, the more accessible (or salient) the entity is. At the extremes are pronouns (these forms typically require a previous mention in the local context) and proper names (these forms are often used without previous mentions of the entity). This type of hierarchy is validated by corpus studies of the distribution of different types of expressions. For instance, pronouns find their antecedents very locally (in a window of 1-2 sentences), while proper names predominantly find theirs at longer distances (Ariel, 1988).[2] Using discourse structure, Asher et al. (2006) show that while anaphoric pronouns systematically obey the right-frontier constraint (i.e., their antecedents have to appear on the right edge of the discourse graph), this is less so for definites, and even less so for proper names.

From a machine learning perspective, these findings suggest that features encoding some aspect of salience (e.g., distance, syntactic context) are likely to receive different sets of parameters depending on the form of the anaphor. This therefore suggests that better parameters are likely to be learned in the

[2]Haghighi and Klein's (2007) generative coreference model mirrors this in the posterior distribution which it assigns to mention types given their salience (see their Table 1).

Type/Count	train	test
3^{rd} pron.	$4,389$	$1,093$
speech pron.	$2,178$	610
proper names	$7,868$	$1,532$
def. NPs	$3,124$	796
others	$1,763$	568
Total	$19,322$	$4,599$

Table 1: Distribution of the different anaphors in ACE

context of different models.[3] While the above studies focus primarily on salience, there are of course other dimensions according to which anaphors differ in their resolution preferences. Thus, the resolution of lexical expressions like definite descriptions and proper names is likely to benefit from the inclusion of features that compare the strings of the anaphor and the candidate antecedent (e.g., string matching) and features that identify particular syntactic configurations like appositive structures. This type of information is however much less likely to help in the resolution of pronominal forms. The problem is that, within a single model, such features are likely to receive strong parameters (due to the fact that they are good predictors for lexical anaphors) in a way that might eventually hurt pronominal resolutions.

Note that our split of referential types only partially cover the referential hierarchies of Ariel (1988) or Gundel et al. (1993). Thus, there is no separate model for demonstrative noun phrases and pronouns: these are very rare in the corpus we used (i.e., the ACE corpus).[4] These expressions were therefore handled through the "others" model. There is however a model for first and second person pronouns (i.e., speech pronouns): this is justified by the fact that these pronouns behave differently from their third person counterparts. These forms indeed often behave like deictics (i.e., they refer to discourse participants) or they appear within a quote.

The total number of anaphors (i.e., of mentions that are not chain heads) in the data is $19,322$ and $4,599$ for training and testing, respectively. The distribution of each anaphoric type is presented in Table 1. Roughly, third person pronouns account for 22-24% of all anaphors in the entire corpus, speech pronouns for 11-13%, proper names for 33-40%, and definite descriptions for 16-17%. The distribution is slightly different from one dataset to another, probably reflecting genre differences. For instance, BNEWS shows a larger proportion of pronouns in general (pronominal forms account for 40-44% of all the anaphoric forms).

Linguistic Form	
pn	α is a proper name {1,0}
def_np	α is a definite description {1,0}
indef_np	α is an indefinite description {1,0}
pro	α is a pronoun {1,0}
Context	
left_pos	POS of the token preceding α
right_pos	POS of the token following α
surr_pos	pair of POS for the tokens surrounding α
Distance	
s_dist	Binned values for sentence distance between π and α
np_dist	Binned values for mention distance between π and α
Morphosyntactic Agreement	
gender	pairs of attributes {masc, fem, neut, unk} for π and α
number	pairs of attributes {sg, pl} for π and α
person	pairs of attributes {1, 2, 3, 4, 5, 6} for π and α
Semantic compatibility	
wn_sense	pairs of Wordnet senses for π and α
String similarity	
str_match	π and α have identical strings {1,0}
left_substr	one mention is a left substring of the other {1,0}
right_substr	one mention is a right substring of the other {1,0}
hd_match	π and α have the same head word {1,0}
Apposition	
apposition	π and α are in an appositive structure {1,0}
Acronym	
acronym	π is an acronym of α or vice versa {1,0}

Table 2: Features used by coreference models.

We use five broad types of features for all mention types, plus three others used by specific types, summarized in Table 3. Our feature extraction relies on limited linguistic processing: we only made use of a sentence detector, a tokenizer, a POS tagger (as provided by the OpenNLP Toolkit[5]) and the WordNet[6] database. Since we did not use parser, lexical heads for the NP mentions were computed using simple heuristics relying solely on POS sequences. Table 2 describes in detail the entire feature set, and Table 3 shows which features were used for which models.

Linguistic form: the referential form of the antecedent candidate: a proper name, a definite de-

[3]Another possible approach would consist in introducing different salience-based features encoding the form of the anaphor.

[4]There are only 114 demonstrative NPs and 12 demonstrative pronouns in the entire ACE training.

[5]`http://opennlp.sf.net.`

[6]`http://wordnet.princeton.edu/`

Features/Types	3P	SP	PN	Def-NP	Oth
Ling. form	√	√	√	√	√
Context	√	√	√	√	√
Distance	√	√	√	√	√
Agreement.	√	√	√	√	√
Sem. compat.	√	√	√	√	√
Str. sim.			√	√	√
Apposition			√	√	
Acronym			√		

Table 3: Features for each type of referential expression.

scription, an indefinite NP, or a pronoun.

Context: the context of the antecedent candidate: these features can be seen as approximations of the grammatical roles, as indicators of the salience of the potential candidate (Grosz et al., 1995). For instance, this includes the part of speech tags surrounding the candidate, as well as a feature that indicates whether the potential antecedent is the first mention in a sentence (approximating subjecthood), and a feature indicating whether the candidate is embedded inside another mention.

Distance: the distance between the anaphor and the candidate, measured by the number of sentences and mentions between them.

Morphosyntactic agreement: indicators of the gender, number, and person of the two mentions. These are determined for non-pronominal NPs with heuristics based on POS tags (e.g., NN vs. NNS for number) and actual mention strings (e.g., whether the mention contains a male/female first name or honorific for gender). These features consist of pairs of attributes, ensuring that not only strict agreement (e.g., *singular-singular*) but also mere compatibility (e.g., *masculine-unknown*) is captured.

Semantic compatibility: features designed to assess whether the two mentions are semantically compatible. For these features, we use the WordNet database: in particular, we collected the synonym set (or synset) as well as the synset of their direct hypernyms associated with each mention. In the case of common nouns, we used the synset associated with the first sense associated with the mention's head word. In the case of proper names, we used the synset associated with the name if available, and the string itself otherwise. For pronouns (which are not part of Wordnet), we simply used the pronominal form.

All these features were used in all five models. While one may question the use of distance for non-pronominal anaphors,[7] their inclusion can be justified in that they might predict some "obviation" effects. Definite descriptions and proper names are sensitive to distance too, although not in the same way as pronouns are: they show a preference for antecedents that appear outside a window of one or two sentences (Ariel, 1988).

Several features are used only for particular mention types:

String similarity: similarity of the anaphor and the candidate strings. Examples are perfect string match, substring matches, and head match (i.e., the two mentions share the same head word).

Appositive: whether the anaphor is an appositive of the antecedent candidate. Since we do not have access to syntactic structure, we use heuristics (e.g., the presence of a comma between the two mentions) to extract this feature.

Acronym: whether the anaphor string is an acronym of the candidate string (or vice versa): e.g., NSF and National Science Foundation.

4 Coreference systems

We evaluate several systems to explore the effect of ranking versus classification and specialized versus monolithic models. The different systems follow a generic architecture. Let $\mathcal{M}$ be the set of mentions present in a document. For all models, each mention $m \in \mathcal{M}$ is associated at test time with a set of antecedent candidates $\mathcal{C}_m$, which includes all the mentions that linearly precede m. The best candidate is determined by the model in use. The final output of each system consists in a list of mention pairs (i.e., the coreference links) which in turn defines (through reflexive, transitive closure) a partition over the set $\mathcal{M}$. Our models are summarized in Table 4.

The use of the discourse status filter is straightforward. For each mention $m \in \mathcal{M}$, the discourse status

[7]In fact, Morton (2000) does not use distance in this case.

Model Name	Model Type	Specialized?	Disc. Status
CLASS	class	No	No
CLASS+DS	class	No	Yes
CLASS+SP	class	Yes	No
CLASS+DS+SP	class	Yes	Yes
RANK+DS+SP	rank	Yes	Yes

Table 4: Model names and their properties.

System	Accuracy
3^{rd} pron.	82.2
speech pron.	66.9
proper names	83.5
def. NPs	66.5
others	63.6

Table 5: Accuracy of the different ranker models.

model is first applied to determine whether m introduces a new discourse entity (i.e., it is classified as "new") or refers back to an existing entity (i.e., it is classified as "old"). If m is classified as "new", the process terminates and goes to the next mention. If m is classified as "old", m along with its set of antecedent candidates $\mathcal{C}_m$ is sent to the model.

For classifiers, we replicate the procedures of Ng and Cardie (2002b). During training, instances are formed by pairing each anaphor with each of its preceding candidates, until the antecedent is reached: the closest preceding antecedent in the case of a pronominal anaphor, or the closest non-pronominal antecedent for other anaphor types. For classifiers, the use of a discourse status filter at test time is optional. When a filter is not used, then a mention is left unresolved if none of the pairs created for a given mention is classified positively. If several pairs for a given mention are classified positively, then the pair with the highest score is selected (i.e., "Best-First" link selection). If a filter is used, then the candidate with the highest score is selected, even if the probability of coreference is less than one-half.[8]

The use of specialized models is simple, for both classifiers and rankers. Specialized models are created for: (i) third person pronouns, (ii) speech pronouns, (iii) proper names, (iv) definite descriptions, (v) other types of phrases. The mention type is determined and the best candidate is chosen by the appropriate model Following Elwell and Baldridge (2008), these models could be interpolated with a monolithic model, or even word specific models, but we have not explored that option here.

The feature sets for the classifiers in the baseline systems includes all the features that were used for the described in Section 3. For the classifiers that do not use specialized models (**CLASS** and **CLASS+DS**), we have also added extra features describing the linguistic form of the potential anaphor (whether it is a pronoun, a proper name, and so on). This is in accordance with standard feature sets in the pairwise approach. It gives these models a chance to learn weights more appropriately for the different types within a single, monolithic model.

5 Experiments

We use the ACE corpus (Phase 2). The corpus has three parts, each corresponding to a different genre: newspaper texts (NPAPER), newswire texts (NWIRE), and broadcast news (BNEWS). Each set is split into a `train` part and a `devtest` part. In our experiments, we consider only *true* ACE mentions.

5.1 Antecedent selection results

We first evaluate the specialized ranker models individually on the task of anaphora resolution: their ability to select a correct antecedent for each anaphor. Following common practice in this task, we report results in terms of *accuracy*, which is simply the ratio of correctly resolved anaphors. The candidate set during testing was formed by taking *all* the mentions that appear before the anaphor. Also, we assume that correctly resolving an anaphor amounts to selecting any of the previous mentions in the entity as the antecedent. The accuracy scores for the different models are presented in Table 5.

[8]This is very similar to the approach of Ng and Cardie (2002a). An important difference is that their system does not necessarily yield an antecedent for each of the anaphors proposed by the discourse status model. In their system, if the coreference classifier finds that none of the candidates for a "new" mention are coreferential, it leaves it unresolved. In this case, the coreference model acts as an additional filter. Not surprisingly, these authors report gains in precision but comparatively larger losses in recall. Our development experiments revealed that forcing a decision on items identified as new provided performed better across all metrics.

The best accuracy results on the entire ACE corpus are found first for the proper name resolver with a score of 83.5%, then for the third person pronoun resolver with 82.2%, then for the definite description and speech pronoun resolvers with 66.9% and 66.5% respectively. The worst scores are obtained for the "others" category. The high scores for the third person pronoun and the proper name rankers most likely follow from the fact that the resolution of these expressions relies on simple, reliable predictors, such as distance and morphosyntactic agreement for pronouns, and string similarity features for proper names. The resolution of definite descriptions and other types of lexical NPs (which are handled through the "others" model) are much more challenging: they rely on lexical semantic and world knowledge, which is only partially encoded via our WordNet-based features. Finally, note that the resolution of speech pronouns is also much harder than that of the other pronominal forms: these expressions are much less (if at all) constrained by recency and agreement. Furthermore, these expressions show a lot of cataphoric uses, which are not considered by our models. The low scores for the "others" category is likely due to the fact that it encompasses very different referential expressions.

5.2 Coreference Results

For evaluating the coreference performance, we rely on three primary metrics: (i) the **link** based MUC metric (Vilain et al., 1995), the **mention** based B^3 metric (Bagga and Baldwin, 1998), and the **entity** based CEAF metric (Luo, 2005). Common to these metrics is: (i) they operate by comparing the set of chains $\mathcal{S}$ produced by the system against the true chains $\mathcal{T}$, and (ii) they report performance in terms of *recall* and *precision*. There are however important differences in how each metric computes these scores, each producing a different bias.

MUC scores are based on the number of *links* (pairs of mentions) common to $\mathcal{S}$ and $\mathcal{T}$. Recall is the number of common links divided by the total number of links in $\mathcal{T}$; precision is the number of common links divided by the total number of links in $\mathcal{S}$. This focus gives MUC two main biases. First, it favors systems that create large chains (and thus fewer entities). For instance, a system that produces a single chain achieves 100% recall without severe degradation in precision. Second, it ignores single mention entities, which are involved in no links.[9]

The B^3 metric was designed to address the MUC metric's shortcomings. It is *mention-based*: it computes both recall and precision scores for each mention i. Let S be the system chain containing m, T be the true chain containing m. The set of correct elements in S is thus $|S \cap T|$. The recall score for a mention i is $\frac{|S \cap T|}{|T|}$, while the precision score for i is $\frac{|S \cap T|}{|S|}$. Overall recall/precision is obtained by averaging over the individual mention scores. The fact that this metric is mention-based by definition solves the problem of single mention entities. Also solved is the bias favoring larger chains, since this will be penalized in the precision score of *each* mention.

The Constrained Entity Aligned F-Measure (CEAF) (Luo, 2005). aligns each system chain S with *at most one* true chain T. It finds the best one-to-one mapping between the set of chains $\mathcal{S}$ and $\mathcal{T}$, which is equivalent to finding the optimal alignment in a bipartite graph. The best mapping maximizes the similarity over pairs of chains (S_i, T_i), where the similarity between two chains is the number of common mentions to the two chains. With CEAF, recall is computed as the total similarity divided by the number of mentions in all the $\mathcal{T}$ (i.e., the self-similarity), while precision is the total similarity divided by the number of mentions in $\mathcal{S}$.

Table 6 gives scores for all three metrics for the different models on the entire ACE corpus. Two main patterns emerge: significant improvements are obtained by using specialized models (**CLASS** vs **CLASS+SP** and **CLASS+DS** vs **CLASS+DS+SP**) and by using a ranker (**CLASS+DS+SP** vs **RANK+DS+SP**). Overall, the **RANK+DS+SP** system significantly outperforms the other systems on the three different metrics.[10]

The f-scores for **RANK+DS+SP** are 71.6% with the MUC metric, 72.7% with the B^3, and 67.0% with the CEAF metric. These scores place the **RANK+DS+SP** among the best coreference resolution systems, since most existing systems are typically under the bar of the 70% in f-score with the

[9]It is worth noting that the MUC corpus does not annotate single mention entities.

[10]Statistical significance was determined with t-tests for both recall and precision scores, with $p < 0.05$.

System	MUC			B^3			CEAF
	R	P	F	R	P	F	F
CLASS	60.8	72.6	66.2	62.4	77.7	69.2	62.3
CLASS+DS	64.9	72.3	68.4	65.6	74.1	69.6	63.4
CLASS+SP	64.8	74.5	69.3	65.3	79.1	71.5	65.0
CLASS+DS+SP	66.8	74.4	70.4	66.4	77.0	71.3	65.3
RANK+DS+SP	67.9	75.7	71.6	66.8	79.8	72.7	67.0

Table 6: Recall (R), Precision (P), and f-score (F) results on the entire ACE corpus using the MUC, B^3, and CEAF metrics. Note that R=P=F for CEAF when using true mentions, as we do here.

MUC and B^3 metrics (Ng, 2005a). An interesting point of comparison is provided by Ng (2007), who also relies on true mentions and reports MUC f-scores only slightly superior to ours (73.8%) while relying on perfect semantic class information. His best results otherwise are 64.6%. The fact that our improvements are consistent across the different evaluation metrics is remarkable, especially given that these three metrics are quite different in the way they compute their scores. The gains in f-score range from 1.2 to 5.4% on the MUC metric (i.e., error reductions of 4 to 15.9%), from 1.4 to 3.5% on the B^3 metric (i.e., error reductions of 4.8 to 11.4%), and from 1.7 to 4.7% on the CEAF metric (i.e., error reductions of 6.9 to 17%). The larger improvements come from recall, with improvements ranging from 1.9 to 7.1% with MUC, from 2.4 to 5.6% with B^3.[11] This suggests that **RANK+DS+SP** predicts many more valid coreference links than the other systems. Smaller but still significant gains are made in precision: **RANK+DS+SP** is also able to reduce the proportion of invalid links.

The overall improvements found with **RANK+DS+SP** suggest that it is able to capitalize on the better antecedent selection capabilities offered by the ranking approach. This is supported by the error analysis on the development data. Errors made by a coreference system can be conceptualized as falling into three main classes: (i) "missed anaphors" (i.e., an anaphoric mention that fails to be linked to a previous mention), (ii) "spurious anaphors" (i.e., an non-anaphoric mention that is linked to a previous mention), and (iii) "invalid resolutions" (i.e., a true anaphor that is linked to a incorrect antecedent). The two first types of error pertain to the determination of the discourse status of the mention, while the third regards the selection of an antecedent (i.e., anaphora resolution). Considering the systems' invalid resolutions, we found that the **RANK+DS+SP** had a much lower error rate: only 17.9% of all true anaphors were incorrectly resolved by this system, against 23.1% for **CLASS**, 24.9% for **CLASS+DS**, 20.4% for **CLASS+SP**, and 22.1% for **CLASS+DS+SP**.

Denis (2007) provides multi-metric scores for the **JOINT-ILP** model of Denis and Baldridge (2007a), which uses integer linear programming for joint inference over coreference resolution and discourse status: f-scores of 73.3%, 68.0%, and 58.9% for MUC, B^3, and CEAF, respectively. Despite the fact that this MUC score beats **RANK+DS+SP**'s, it is actually *worse* than even the basic model **CLASS** for B^3 and CEAF. This difference fact that MUC gives more recall credit for large chains without a consequent precision reduction, and shows the importance of using B^3 and CEAF scores in addition to MUC.

Denis (2007) also extends the **JOINT-ILP** system by adding named entity resolution and constraints on transitivity with respect to coreference links. The best model reported there (**JOINT-DS-NE-AE-ILP**) obtains f-scores of 70.1%, 72.7%, and 66.2% for MUC, B^3, and CEAF, respectively. Interestingly, **RANK+DS+SP** actually performs better across all metrics despite being a simpler model with fewer sources of information.

5.3 Oracle results

Using specialized rankers with a discourse status classifier yields coreference performance superior to that given by various classification-based baseline systems. Crucially, these improvements have been

[11]Recall that recall and precision scores are identical with CEAF, due to the fact that we are using true mention boundaries.

System	MUC			B^3			CEAF
	R	P	F	R	P	F	F
RANK+DS+SP	67.9	75.7	71.6	66.8	79.8	72.7	67.0
RANK+DS-ORACLE+SP	79.1	79.1	79.1	75.4	76.0	75.7	76.9
LINK-ORACLE	78.8	100.0	88.1	74.3	100.0	85.2	79.7

Table 7: Recall (R), Precision (P), and f-score (F) results for **RANK+DS-ORACLE+SP** and **LINK-ORACLE** on the entire ACE corpus.

possible using a discourse status model that has an accuracy of just 80.8%. Clearly, the performance of the discourse status module has a direct impact on the performance of the entire coreference system. On the one hand, misclassified anaphors are simply not resolved by the rankers: this limits the recall of the coreference system. On the other hand, misclassified non-anaphors are linked to a previous mention: this limits precision.

In order to further assess the impact of the errors made by the discourse status classifier, we build two different oracle systems. The first oracle system, **RANK+DS-ORACLE+SP**, uses the specialized rankers in combination with a perfect discourse status classifier. That is, this system knows for each mention whether it is anaphoric or not: the only errors made by such a system are invalid resolutions. **RANK+DS-ORACLE+SP** thus provides an upperbound for the **RANK+DS+SP** model. The results for this oracle are given in Table 7: they show substantial improvements over **RANK+DS+SP**, which suggests that the **RANK+DS+SP** has also the potential to be further improved if used in combination with a more accurate discourse status classifier.

The second oracle system, **LINK-ORACLE**, uses the discourse status classifier with a perfect coreference resolver. That is, this system has perfect knowledge regarding the antecedents of anaphors: the errors made by such a system are only errors in the discourse status of mentions. The results for **LINK-ORACLE** are also reported in Table 7. These figures show that however accurate our models are at picking a correct antecedent for a true anaphor, the best they can achieve in terms of f-scores is 88.1% with MUC, 85.2% with B^3, and 79.7% with CEAF.

6 Conclusion

We present and evaluate two straight-forward tactics for improving coreference resolution: (i) ranking models, and (ii) separate, specialized models for different types of referring expressions. The specialized rankers are used in combination with a discourse status classifier which determines the mentions that are sent to the rankers. This simple pipeline architecture produces significant improvements over various implementations of the standard, classifier-based coreference system. In turn, these strategies could be integrated with the joint inference models we have explored elsewhere (Denis and Baldridge, 2007a; Denis, 2007) and which have obtained performance improvements that are orthogonal to those obtained here.

This paper's improvements are consistent across the three main coreference evaluation metrics: MUC, B^3, and CEAF.[12] We attribute improvements to: (i) the better antecedent selection capabilities offered by the ranking approach, and (ii) the division of labor between specialized models, allowing each one to better model the corresponding distribution.

Acknowledgments

We would like to thank Nicholas Asher, Andy Kehler, Ray Mooney, and the three anonymous reviewers for their comments. This work was supported by NSF grant IIS-0535154.

References

M. Ariel. 1988. Referring and accessibility. *Journal of Linguistics*, pages 65–87.

N. Asher, P. Denis, and B. Reese. 2006. Names and pops and discourse structure. In *Workshop on Constraints in Discourse*, Maynooth, Ireland.

A. Bagga and B. Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of LREC 1998*, pages 563–566.

[12]We *strongly* advocate that coreference results should ***never*** be presented in terms of MUC scores alone.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of ACL 2005*, Ann Arbor, Michigan.

M. Collins and N. Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures and the voted perceptron. In *Proceedings of ACL 2002*, pages 263–270, Philadelphia, PA.

H. Daumé III and D. Marcu. 2005. A large-scale exploration of effective global features for a joint entity detection and tracking model. In *Proceedings of HLT-EMNLP 2005*, Vancouver, Canada.

P. Denis and J. Baldridge. 2007a. Joint determination of anaphoricity and coreference resolution using integer programming. In *Proceedings of HLT-NAACL 2007*, Rochester, NY.

P. Denis and J. Baldridge. 2007b. A ranking approach to pronoun resolution. In *Proceedings of IJCAI 2007*, Hyderabad, India.

Pascal Denis. 2007. *New Learning Models for Robust Reference Resolution*. Ph.D. thesis, The University of Texas at Austin.

R. Elwell and J. Baldridge. 2008. Discourse connective argument identification with connective specific rankers. In *Proceedings of the International Conference on Semantic Computing*, Santa Clara, CA.

B. Grosz, A. Joshi, and S. Weinstein. 1995. Centering: A framework for modelling the local coherence of discourse. *Computational Linguistics*, 2(21).

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

A. Haghighi and D. Klein. 2007. Unsupervised coreference resolution in a nonparametric Bayesian model. In *Proceedings ACL 2007*, pages 848–855, Prague, Czech Republic.

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 2004*.

X. Luo. 2005. On coreference resolution performance metrics. In *Proceedings of HLT-NAACL 2005*, pages 25–32.

R. 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.

A. McCallum and B. Wellner. 2004. Conditional models of identity uncertainty with application to noun coreference. In *Proceedings of NIPS 2004*.

J. F. McCarthy and W. G. Lehnert. 1995. Using decision trees for coreference resolution. In *IJCAI*, pages 1050–1055.

T. Morton. 2000. Coreference for NLP applications. In *Proceedings of ACL 2000*, Hong Kong.

V. Ng and C. Cardie. 2002a. Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. In *Proceedings of COLING 2002*.

V. Ng and C. Cardie. 2002b. Improving machine learning approaches to coreference resolution. In *Proceedings of ACL 2002*, pages 104–111.

V. Ng. 2004. Learning noun phrase anaphoricity to improve coreference resolution: Issues in representation and optimization. In *Proceedings of ACL 2004*.

V. Ng. 2005a. Machine learning for coreference resolution: From local classification to global ranking. In *Proceedings of ACL 2005*, pages 157–164, Ann Arbor, MI.

V. Ng. 2005b. Supervised ranking for pronoun resolution: Some recent improvements. In *Proceedings of AAAI 2005*.

V. Ng. 2007. Semantic class induction and coreference resolution. In *Proceedings of ACL 2007*.

M. Osborne and J. Baldridge. 2004. Ensemble-based active learning for parse selection. In *Proceedings of HLT-NAACL 2004*, pages 89–96, Boston, MA.

E. F. Prince. 1981. Toward a taxonomy of given-new information. In P. Cole, editor, *Radical Pragmatics*, pages 223–255. Academic Press, New York.

D. Ravichandran, E. Hovy, and F. J. Och. 2003. Statistical QA - classifier vs re-ranker: What's the difference? In *Proceedings of the ACL Workshop on Multilingual Summarization and Question Answering–Machine Learning and Beyond*.

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

K. Toutanova, P. Markova, and C. Manning. 2004. The leaf projection path view of parse trees: Exploring string kernels for HPSG parse selection. In *Proceedings of EMNLP 2004*, pages 166–173, Barcelona.

O. Uryupina. 2004. Linguistically motivated sample selection for coreference resolution. In *Proceedings of DAARC 2004*, Furnas.

R. van der Sandt. 1992. Presupposition projection as anaphora resolution. *Journal of Semantics*, 9:333–377.

M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. 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.

X. Yang, G. Zhou, J. Su, and C.L. Tan. 2003. Coreference resolution using competitive learning approach. In *Proceedings of ACL 2003*, pages 176–183.

Learning with Probabilistic Features for Improved Pipeline Models

Razvan C. Bunescu
School of EECS
Ohio University
Athens, OH 45701
bunescu@ohio.edu

Abstract

We present a novel learning framework for pipeline models aimed at improving the communication between consecutive stages in a pipeline. Our method exploits the confidence scores associated with outputs at any given stage in a pipeline in order to compute probabilistic features used at other stages downstream. We describe a simple method of integrating probabilistic features into the linear scoring functions used by state of the art machine learning algorithms. Experimental evaluation on dependency parsing and named entity recognition demonstrate the superiority of our approach over the baseline pipeline models, especially when upstream stages in the pipeline exhibit low accuracy.

1 Introduction

Machine learning algorithms are used extensively in natural language processing. Applications range from fundamental language tasks such as part of speech (POS) tagging or syntactic parsing, to higher level applications such as information extraction (IE), semantic role labeling (SRL), or question answering (QA). Learning a model for a particular language processing problem often requires the output from other natural language tasks. Syntactic parsing and dependency parsing usually start with a textual input that is tokenized, split in sentences and POS tagged. In information extraction, named entity recognition (NER), coreference resolution, and relation extraction (RE) have been shown to benefit from features that use POS tags and syntactic dependencies. Similarly, most SRL approaches assume a parse tree representation of the input sentences. The common practice in modeling such dependencies is to use a pipeline organization, in which the output of one task is fed as input to the next task in the sequence. One advantage of this model is that it is very simple to implement; it also allows for a modular approach to natural language processing. The key disadvantage is that errors propagate between stages in the pipeline, significantly affecting the quality of the final results. One solution is to solve the tasks jointly, using the principled framework of probabilistic graphical models. Sutton et al. (2004) use factorial Conditional Random Fields (CRFs) (Lafferty et al., 2001) to jointly predict POS tags and segment noun phrases, improving on the cascaded models that perform the two tasks in sequence. Wellner et al. (2004) describe a CRF model that integrates the tasks of citation segmentation and citation matching. Their empirical results show the superiority of the integrated model over the pipeline approach. While more accurate than their pipeline analogues, probabilistic graphical models that jointly solve multiple natural language tasks are generally more demanding in terms of finding the right representations, the associated inference algorithms and their computational complexity. Recent negative results on the integration of syntactic parsing with SRL (Sutton and McCallum, 2005) provide additional evidence for the difficulty of this general approach. When dependencies between the tasks can be formulated in terms of constraints between their outputs, a simpler approach is to solve the tasks separately and integrate the constraints in a linear programming formulation, as proposed by Roth and

Yih (2004) for the simultaneous learning of named entities and relations between them. More recently, Finkel et al. (2006) model the linguistic pipelines as Bayesian networks on which they perform Monte Carlo inference in order to find the most likely output for the final stage in the pipeline.

In this paper, we present a new learning method for pipeline models that mitigates the problem of error propagation between the tasks. Our method exploits the probabilities output by any given stage in the pipeline as weights for the features used at other stages downstream. We show a simple method of integrating probabilistic features into linear scoring functions, which makes our approach applicable to state of the art machine learning algorithms such as CRFs and Support Vector Machines (Vapnik, 1998; Schölkopf and Smola, 2002). Experimental results on dependency parsing and named entity recognition show useful improvements over the baseline pipeline models, especially when the basic pipeline components exhibit low accuracy.

2 Learning with Probabilistic Features

We consider that the task is to learn a mapping from inputs $x \in \mathcal{X}$ to outputs $y \in \mathcal{Y}(x)$. Each input x is also associated with a different set of outputs $z \in \mathcal{Z}(x)$ for which we are given a probabilistic confidence measure $p(z|x)$. In a pipeline model, z would correspond to the annotations performed on the input x by all stages in the pipeline other than the stage that produces y. For example, in the case of dependency parsing, x is a sequence of words, y is a set of word-word dependencies, z is a sequence of POS tags, and $p(z|x)$ is a measure of the confidence that the POS tagger has in the output z. Let ϕ be a representation function that maps an example (x, y, z) to a feature vector $\phi(x, y, z) \in \mathbb{R}^d$, and $w \in \mathbb{R}^d$ a parameter vector. Equations (1) and (2) below show the traditional method for computing the optimal output $\hat{y}$ in a pipeline model, assuming a linear scoring function defined by w and ϕ.

$$\hat{y}(x) = \operatorname*{argmax}_{y \in \mathcal{Y}(x)} w \cdot \phi(x, y, \hat{z}(x)) \quad (1)$$

$$\hat{z}(x) = \operatorname*{argmax}_{z \in \mathcal{Z}(x)} p(z|x) \quad (2)$$

The weight vector w is learned by optimizing a predefined objective function on a training dataset.

In the model above, only the best annotation $\hat{z}$ produced by upstream stages is used for determining the optimal output $\hat{y}$. However, $\hat{z}$ may be an incorrect annotation, while the correct annotation may be ignored because it was assigned a lower confidence value. We propose exploiting all possible annotations and their probabilities as illustrated in the new model below:

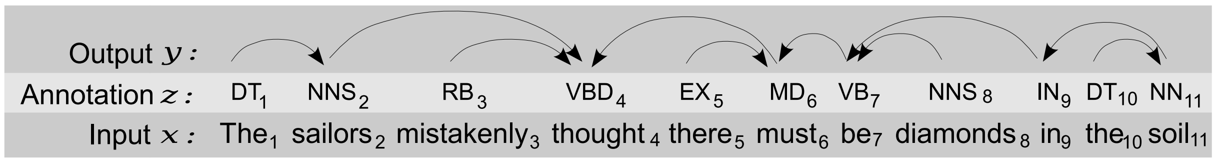
$$\hat{y}(x) = \operatorname*{argmax}_{y \in \mathcal{Y}(x)} w \cdot \psi(x, y) \quad (3)$$

$$\psi(x, y) = \sum_{z \in \mathcal{Z}(x)} p(z|x) \cdot \phi(x, y, z) \quad (4)$$

In most cases, directly computing $\psi(x, y)$ is unfeasible, due to a large number of annotations in $\mathcal{Z}(x)$. In our dependency parsing example, $\mathcal{Z}(x)$ contains all possible POS taggings of sentence x; consequently its cardinality is exponential in the length of the sentence. A more efficient way of computing $\psi(x, y)$ can be designed based on the observation that most components ϕ_i of the original feature vector ϕ utilize only a limited amount of evidence from the example (x, y, z). We define $(\tilde{x}, \tilde{y}, \tilde{z}) \in \mathcal{F}_i(x, y, z)$ to capture the actual evidence from (x, y, z) that is used by one instance of feature function ϕ_i. We call $(\tilde{x}, \tilde{y}, \tilde{z})$ a *feature instance* of ϕ_i in the example (x, y, z). Correspondingly, $\mathcal{F}_i(x, y, z)$ is the set of all feature instances of ϕ_i in example (x, y, z). Usually, $\phi_i(x, y, z)$ is set to be equal with the number of instances of ϕ_i in example (x, y, z), i.e. $\phi_i(x, y, z) = |\mathcal{F}_i(x, y, z)|$. Table 1 illustrates three feature instances $(\tilde{x}, \tilde{y}, \tilde{z})$ generated by three typical dependency parsing features in the example from Figure 1. Because the same feature may be instantiated multiple times in the same example, the components of each feature instance are annotated with their positions relative to the example. Given these definitions, the feature vector $\psi(x, y)$ from (4) can be

	ϕ_1 : **DT → NN**	ϕ_2 : **NNS → thought**	ϕ_3 : **be ← in**
$\tilde{y}$	$10 \rightarrow 11$	$2 \rightarrow 4$	$7 \leftarrow 9$
$\tilde{z}$	DT$_{10}$ NN$_{11}$	NNS$_2$	
$\tilde{x}$		thought$_4$	be$_7$ in$_9$
$\lvert\mathcal{F}_i\rvert$	$O(\lvert x\rvert^2)$	$O(\lvert x\rvert)$	$O(1)$

Table 1: Feature instances.



Output y :
Annotation z :
DT$_1$ NNS$_2$ RB$_3$ VBD$_4$ EX$_5$ MD$_6$ VB$_7$ NNS$_8$ IN$_9$ DT$_{10}$ NN$_{11}$
Input x :
The$_1$ sailors$_2$ mistakenly$_3$ thought$_4$ there$_5$ must$_6$ be$_7$ diamonds$_8$ in$_9$ the$_{10}$ soil$_{11}$

Figure 1: Dependency Parsing Example.

rewritten in a component-wise manner as follows:

$$\begin{aligned}
\psi(x,y) &= [\psi_1(x,y)\ \ldots\ \psi_d(x,y)] \qquad (5)\\
\psi_i(x,y) &= \sum_{z\in\mathcal{Z}(x)} p(z|x)\cdot\phi_i(x,y,z)\\
&= \sum_{z\in\mathcal{Z}(x)} p(z|x)\cdot|\mathcal{F}_i(x,y,z)|\\
&= \sum_{z\in\mathcal{Z}(x)} p(z|x) \sum_{(\tilde{x},\tilde{y},\tilde{z})\in\mathcal{F}_i(x,y,z)} 1\\
&= \sum_{z\in\mathcal{Z}(x)} \sum_{(\tilde{x},\tilde{y},\tilde{z})\in\mathcal{F}_i(x,y,z)} p(z|x)\\
&= \sum_{(\tilde{x},\tilde{y},\tilde{z})\in\mathcal{F}_i(x,y,\mathcal{Z}(x))} \sum_{z\in\mathcal{Z}(x),z\supseteq\tilde{z}} p(z|x)
\end{aligned}$$

where $\mathcal{F}_i(x,y,\mathcal{Z}(x))$ stands for:

$$\mathcal{F}_i(x,y,\mathcal{Z}(x)) = \bigcup_{z\in\mathcal{Z}(x)} \mathcal{F}_i(x,y,z)$$

We introduce $p(\tilde{z}|x)$ to denote the expectation:

$$p(\tilde{z}|x) = \sum_{z\in\mathcal{Z}(x),z\supseteq\tilde{z}} p(z|x)$$

Then $\psi_i(x,y)$ can be written compactly as:

$$\psi_i(x,y) = \sum_{(\tilde{x},\tilde{y},\tilde{z})\in\mathcal{F}_i(x,y,\mathcal{Z}(x))} p(\tilde{z}|x) \qquad (6)$$

The total number of terms in (6) is equal with the number of instantiations of feature ϕ_i in the example (x,y) across all possible annotations $z\in\mathcal{Z}(x)$, i.e. $|\mathcal{F}_i(x,y,\mathcal{Z}(x))|$. Usually this is significantly smaller than the exponential number of terms in (4). The actual number of terms depends on the particular feature used to generate them, as illustrated in the last row of Table 1 for the three features used in dependency parsing. The overall time complexity for calculating $\psi(x,y)$ also depends on the time complexity needed to compute the expectations $p(\tilde{z}|x)$. When z is a sequence, $p(\tilde{z}|x)$ can be computed efficiently using a constrained version of the forward-backward algorithm (to be described in Section 3). When z is a tree, $p(\tilde{z}|x)$ will be computed using a constrained version of the CYK algorithm (to be described in Section 4).

The time complexity can be further reduced if instead of $\psi(x,y)$ we use its subcomponent $\hat{\psi}(x,y)$ that is calculated based only on instances that appear in the optimal annotation $\hat{z}$:

$$\begin{aligned}
\hat{\psi}(x,y) &= [\hat{\psi}_1(x,y)\ \ldots\ \hat{\psi}_d(x,y)] \qquad (7)\\
\hat{\psi}_i(x,y) &= \sum_{(\tilde{x},\tilde{y},\tilde{z})\in\mathcal{F}_i(x,y,\hat{z})} p(\tilde{z}|x) \qquad (8)
\end{aligned}$$

The three models are summarized in Table 2 below. In the next two sections we illustrate their application to two common tasks in language processing: dependency parsing and named entity recognition.

M_1	$\hat{y}(x) = \operatorname{argmax}_{y\in\mathcal{Y}(x)} w\cdot\phi(x,y)$ $\phi(x,y) = \phi(x,y,\hat{z}(x))$ $\hat{z}(x) = \operatorname{argmax}_{z\in\mathcal{Z}(x)} p(z\|x)$
M_2	$\hat{y}(x) = \operatorname{argmax}_{y\in\mathcal{Y}(x)} w\cdot\psi(x,y)$ $\psi(x,y) = [\psi_1(x,y)\ \ldots\ \psi_d(x,y)]$ $\psi_i(x,y) = \sum_{(\tilde{x},\tilde{y},\tilde{z})\in\mathcal{F}_i(x,y,\mathcal{Z}(x))} p(\tilde{z}\|x)$
M_3	$\hat{y}(x) = \operatorname{argmax}_{y\in\mathcal{Y}(x)} w\cdot\hat{\psi}(x,y)$ $\hat{\psi}(x,y) = [\hat{\psi}_1(x,y)\ \ldots\ \hat{\psi}_d(x,y)]$ $\hat{\psi}_i(x,y) = \sum_{(\tilde{x},\tilde{y},\tilde{z})\in\mathcal{F}_i(x,y,\hat{z})} p(\tilde{z}\|x)$

Table 2: Three Pipeline Models.

3 Dependency Parsing Pipeline

In a traditional dependency parsing pipeline (model M_1 in Table 2), an input sentence x is first augmented with a POS tagging $\hat{z}(x)$, and then processed by a dependency parser in order to obtain a dependency structure $\hat{y}(x)$. To evaluate the new pipeline models we use MSTPARSER[1], a linearly scored dependency parser developed by McDonald et al. (2005). Following the edge based factorization method of Eisner (1996), the score of a dependency tree in the first order version is defined as the sum of the scores of all edges in the tree. Equivalently, the feature vector of a dependency tree is defined as the sum of the feature vectors of all edges in the tree:

$$M_1\text{: } \phi(x,y) = \sum_{u \to v \in y} \phi(x, u \to v, \hat{z}(x))$$

$$M_2\text{: } \psi(x,y) = \sum_{u \to v \in y} \psi(x, u \to v)$$

$$M_3\text{: } \hat{\psi}(x,y) = \sum_{u \to v \in y} \hat{\psi}(x, u \to v)$$

For each edge $u \to v \in y$, MSTPARSER generates features based on a set of feature templates that take into account the words and POS tags at positions u, v, and their left and right neighbors $u \pm 1$, $v \pm 1$. For example, a particular feature template $\mathcal{T}$ used inside MSTPARSER generates the following POS bigram features:

$$\phi_i(x, u \to v, z) = \begin{cases} 1, & \text{if } \langle z_u, z_v \rangle = \langle t_1, t_2 \rangle \\ 0, & \text{otherwise} \end{cases}$$

where $t_1, t_2 \in \mathcal{P}$ are the two POS tags associated with feature index i. By replacing y with $u \to v$ in the feature expressions from Table 2, we obtain the following formulations:

$$M_1\text{:} \phi_i(x, u \to v) = \begin{cases} 1, & \text{if } \langle \hat{z}_u, \hat{z}_v \rangle = \langle t_1, t_2 \rangle \\ 0, & \text{otherwise} \end{cases}$$

$$M_2\text{:} \psi_i(x, u \to v) = p(\tilde{z} = \langle t_1, t_2 \rangle | x)$$

$$M_3\text{:} \hat{\psi}_i(x, u \to v) = \begin{cases} p(\tilde{z} = \langle t_1, t_2 \rangle | x), & \text{if } \langle \hat{z}_u, \hat{z}_v \rangle = \langle t_1, t_2 \rangle \\ 0, & \text{otherwise} \end{cases}$$

where, following the notation from Section 2, $\tilde{z} = \langle z_u, z_v \rangle$ is the actual evidence from z that is used by feature i, and $\hat{z}$ is the top scoring annotation produced by the POS tagger. The implementation in MSTPARSER corresponds to the traditional pipeline model M_1. Given a method for computing feature probabilities $p(\tilde{z} = \langle t_1, t_2 \rangle | x)$, it is straightforward to modify MSTPARSER to implement models M_2 and M_3 – we simply replace the feature vectors ϕ with ψ and $\hat{\psi}$ respectively. As mentioned in Section 2, the time complexity of computing the feature vectors ψ in model M_2 depends on the complexity of the actual evidence $\tilde{z}$ used by the features. For example, the feature template $\mathcal{T}$ used above is based on the POS tags at both ends of a dependency edge, consequently it would generate $|\mathcal{P}|^2$ features in model M_2 for any given edge $u \to v$. There are however feature templates used in MSTPARSER that are based on the POS tags of up to 4 tokens in the input sentence, which means that for each edge they would generate $|\mathcal{P}|^4 \approx 4.5M$ features. Whether using all these probabilistic features is computationally feasible or not also depends on the time complexity of computing the confidence measure $p(\tilde{z}|x)$ associated with each feature.

3.1 Probabilistic POS features

The new pipeline models M_2 and M_3 require an annotation model that, at a minimum, facilitates the computation of probabilistic confidence values for each output. We chose to use linear chain CRFs (Lafferty et al., 2001) since CRFs can be easily modified to compute expectations of the type $p(\tilde{z}|x)$, as needed by M_2 and M_3.

The CRF tagger was implemented in MALLET (McCallum, 2002) using the original feature templates from (Ratnaparkhi, 1996). The model was trained on sections 2–21 from the English Penn Treebank (Marcus et al., 1993). When tested on section 23, the CRF tagger obtains 96.25% accuracy, which is competitive with more finely tuned systems such as Ratnaparkhi's MaxEnt tagger.

We have also implemented in MALLET a constrained version of the forward-backward procedure that allows computing feature probabilities $p(\tilde{z}|x)$. If $\tilde{z} = \langle t_{i_1} t_{i_2} ... t_{i_k} \rangle$ specifies the tags at k positions in the sentence, then the procedure recomputes the α parameters for all positions between i_1 and i_k by constraining the state transitions to pass through the specified tags at the k positions. A similar approach was used by Culotta et al. in (2004) in order to associate confidence values with sequences of contiguous tokens identified by a CRF model as fields in an information extraction task. The constrained proce-

[1]URL: http://sourceforge.net/projects/mstparser

dure requires $(i_k - i_1)|\mathcal{P}|^2 = O(N|\mathcal{P}|^2)$ multiplications in an order 1 Markov model, where N is the length of the sentence. Because MSTPARSER uses an edge based factorization of the scoring function, the constrained forward procedure will need to be run for each feature template, for each pair of tokens in the input sentence x. If the evidence $\tilde{z}$ required by the feature template $\mathcal{T}$ constrains the tags at k positions, then the total time complexity for computing the probabilistic features $p(\tilde{z}|x)$ generated by $\mathcal{T}$ is:

$$O(N^3|\mathcal{P}|^{k+2}) = O(N|\mathcal{P}|^2) \cdot O(N^2) \cdot O(|\mathcal{P}|^k) \quad (9)$$

As mentioned earlier, some feature templates used in the dependency parser constrain the POS tags at 4 positions, leading to a $O(N^3|\mathcal{P}|^6)$ time complexity for a length N sentence. Experimental runs on the same machine that was used for CRF training show that such a time complexity is not yet feasible, especially because of the large size of $\mathcal{P}$ (46 POS tags). In order to speed up the computation of probabilistic features, we made the following two approximations:

1. Instead of using the constrained forward-backward procedure, we enforce an independence assumption between tags at different positions and rewrite $p(\tilde{z} = \langle t_{i_1} t_{i_2} ... t_{i_k} \rangle | x)$ as:

   $$p(t_{i_1} t_{i_2} ... t_{i_k} | x) \approx \prod_{j=1}^{k} p(t_{i_j}|x)$$

   The marginal probabilities $p(t_{i_j}|x)$ are easily computed using the original forward and backward parameters as:

   $$p(t_{i_j}|x) = \frac{\alpha_{i_j}(t_{i_j}|x)\beta_{i_j}(t_{i_j}|x)}{Z(x)}$$

   This approximation eliminates the factor $O(N|\mathcal{P}|^2)$ from the time complexity in (9).

2. If any of the marginal probabilities $p(t_{i_j}|x)$ is less than a predefined threshold $(\tau|\mathcal{P}|)^{-1}$, we set $p(\tilde{z}|x)$ to 0. When $\tau \geq 1$, the method is guaranteed to consider at least the most probable state when computing the probabilistic features. Looking back at Equation (4), this is equivalent with summing feature vectors only over the most probable annotations $z \in \mathcal{Z}(x)$.

   The approximation effectively replaces the factor $O(|\mathcal{P}|^k)$ in (9) with a quasi-constant factor.

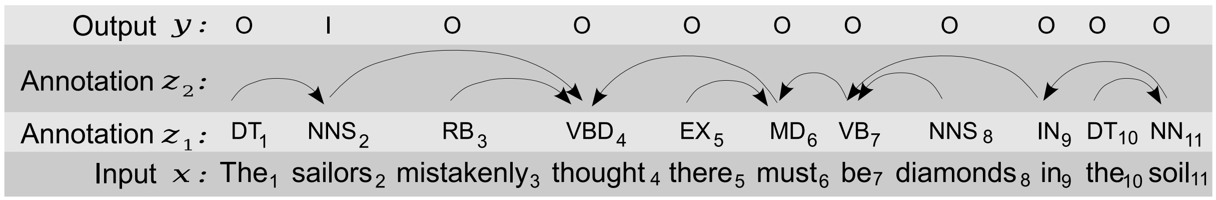
The two approximations lead to an overall time complexity of $O(N^2)$ for computing the probabilistic features associated with any feature template $\mathcal{T}$, plus $O(N|\mathcal{P}|^2)$ for the unconstrained forward-backward procedure. We will use M_2' to refer to the model M_2 that incorporates the two approximations. The independence assumption from the first approximation can be relaxed without increasing the asymptotic time complexity by considering as independent only chunks of contiguous POS tags that are at least a certain number of tokens apart. Consequently, the probability of the tag sequence will be approximated with the product of the probabilities of the tag chunks, where the exact probability of each chunk is computed in constant time with the constrained forward-backward procedure. We will use M_2'' to refer to the resulting model.

3.2 Experimental Results

MSTPARSER was trained on sections 2–21 from the WSJ Penn Treebank, using the gold standard POS tagging. The parser was then evaluated on section 23, using the POS tagging output by the CRF tagger. For model M_1 we need only the best output from the POS tagger. For models M_2' and M_2'' we compute the probability associated with each feature using the corresponding approximations, as described in the previous section. In model M_2'' we consider as independent only chunks of POS tags that are 4 tokens or more apart. If the distance between the chunks is less than 4 tokens, the probability for the entire tag sequence in the feature is computed exactly using the constrained forward-backward procedure. Table 3 shows the accuracy obtained by models M_1, $M_2'(\tau)$ and $M_2''(\tau)$ for various values of the threshold parameter τ. The accuracy is computed over unlabeled dependencies i.e. the percentage of words for which the parser has correctly identified the parent in the dependency tree. The pipeline

M_1	$M_2'(1)$	$M_2'(2)$	$M_2'(4)$	$M_2''(4)$
88.51	88.66	88.67	88.67	**88.70**

Table 3: Dependency parsing results.



Output y: O I O O O O O O O O O
Annotation z_2:
Annotation z_1: DT1 NNS2 RB3 VBD4 EX5 MD6 VB7 NNS8 IN9 DT10 NN11
Input x: The1 sailors2 mistakenly3 thought4 there5 must6 be7 diamonds8 in9 the10 soil11

Figure 2: Named Entity Recognition Example.

model M_2' that uses probabilistic features outperforms the traditional pipeline model M_1. As expected, M_2'' performs slightly better than M_2', due to a more exact computation of feature probabilities. Overall, only by using the probabilities associated with the POS features, we achieve an absolute error reduction of 0.19%, in a context where the POS stage in the pipeline already has a very high accuracy of 96.25%. We expect probabilistic features to yield a more substantial improvement in cases where the pipeline model contains less accurate upstream stages. Such a case is that of NER based on a combination of POS and dependency parsing features.

4 Named Entity Recognition Pipeline

In Named Entity Recognition (NER), the task is to identify textual mentions of predefined types of entities. Traditionally, NER is modeled as a sequence classification problem: each token in the input sentence is tagged as being either inside (I) or outside (O) of an entity mention. Most sequence tagging approaches use the words and the POS tags in a limited neighborhood of the current sentence position in order to compute the corresponding features. We augment these *flat features* with a set of *tree features* that are computed based on the words and POS tags found in the proximity of the current token in the dependency tree of the sentence. We argue that such dependency tree features are better at capturing predicate-argument relationships, especially when they span long stretches of text. Figure 2 shows a sentence x together with its POS tagging z_1, dependency links z_2, and an output tagging y. Assuming the task is to recognize mentions of people, the word *sailors* needs to be tagged as inside. If we extracted only flat features using a symmetric window of size 3, the relationship between *sailors* and *thought* would be missed. This relationship is useful, since an agent of the predicate *thought* is likely to be a person entity. On the other hand, the nodes *sailors* and *thought* are adjacent in the dependency tree of the sentence. Therefore, their relationship can be easily captured as a dependency tree feature using the same window size.

For every token position, we generate flat features by considering all unigrams, bigrams and trigrams that start with the current token and extend either to the left or to the right. Similarly, we generate tree features by considering all unigrams, bigrams and trigrams that start with the current token and extend in any direction in the undirected version of the dependency tree. The tree features are also augmented with the actual direction of the dependency arcs between the tokens. If we use only words to create n-gram features, the token *sailors* will be associated with the following features:

- **Flat:** *sailors*, *the sailors*, ⟨*S*⟩ *the sailors*, *sailors mistakenly*, *sailors mistakenly thought*.
- **Tree:** *sailors*, *sailors* $\leftarrow$ *the*, *sailors* $\rightarrow$ *thought*, *sailors* $\rightarrow$ *thought* $\leftarrow$ *must*, *sailors* $\rightarrow$ *thought* $\leftarrow$ *mistakenly*.

We also allow n-grams to use word classes such as POS tags and any of the following five categories: ⟨1C⟩ for tokens consisting of one capital letter, ⟨AC⟩ for tokens containing only capital letters, ⟨FC⟩ for tokens that start with a capital letter, followed by small letters, ⟨CD⟩ for tokens containing at least one digit, and ⟨CRT⟩ for the current token.

The set of features can then be defined as a Cartesian product over word classes, as illustrated in Figure 3 for the original tree feature *sailors* $\rightarrow$ *thought* $\leftarrow$ *mistakenly*. In this case, instead of one completely lexicalized feature, the model will consider 12 different features such as *sailors* $\rightarrow$ *VBD* $\leftarrow$ *RB*, *NNS* $\rightarrow$ *thought* $\leftarrow$ *RB*, or *NNS* $\rightarrow$ *VBD* $\leftarrow$ *RB*.

$$\begin{bmatrix} \langle \text{CRT} \rangle \\ \text{NNS} \\ \text{sailors} \end{bmatrix} \times [\rightarrow] \times \begin{bmatrix} \text{VBD} \\ \text{thought} \end{bmatrix} \times [\leftarrow] \times \begin{bmatrix} \text{RB} \\ \text{mistakenly} \end{bmatrix}$$

Figure 3: Dependency tree features.

The pipeline model M_2 uses features that appear in all possible annotations $z = \langle z_1, z_2 \rangle$, where z_1 and z_2 are the POS tagging and the dependency parse respectively. If the corresponding evidence is $\tilde{z} = \langle \tilde{z}_1, \tilde{z}_2 \rangle$, then:

$$p(\tilde{z}|x) = p(\tilde{z}_2|\tilde{z}_1, x)p(\tilde{z}_1|x)$$

For example, $\textit{NNS}_2 \rightarrow \textit{thought}_4 \leftarrow \textit{RB}_3$ is a feature instance for the token *sailors* in the annotations from Figure 2. This can be construed as having been generated by a feature template $\mathcal{T}$ that outputs the POS tag t_i at the current position, the word x_j that is the parent of x_i in the dependency tree, and the POS tag t_k of another dependent of x_j (i.e. $t_i \rightarrow x_j \leftarrow t_k$). The probability $p(\tilde{z}|x)$ for this type of features can then be written as:

$$p(\tilde{z}|x) = p(i \rightarrow j \leftarrow k | t_i, t_k, x) \cdot p(t_i, t_k|x)$$

The two probability factors can be computed exactly as follows:

1. The M_2 model for dependency parsing from Section 3 is used to compute the probabilistic features $\psi(x, u \rightarrow v | t_i, t_k)$ by constraining the POS annotations to pass through tags t_i and t_k at positions i and k. The total time complexity for this step is $O(N^3|\mathcal{P}|^{k+2})$.

2. Having access to $\psi(x, u \rightarrow v | t_i, t_k)$, the factor $p(i \rightarrow j \leftarrow k | t_i, t_k, x)$ can be computed in $O(N^3)$ time using a constrained version of Eisner's algorithm, as will be explained in Section 4.1.

3. As described in Section 3.1, computing the expectation $p(t_i, t_k|x)$ takes $O(N|\mathcal{P}^2|)$ time using the constrained forward-backward algorithm.

The current token position i can have a total of N values, while j and k can be any positions other than i. Also, t_i and t_k can be any POS tag from $\mathcal{P}$. Consequently, the feature template $\mathcal{T}$ induces $O(N^3|\mathcal{P}|^2)$ feature instances. Overall, the time complexity for computing the feature instances generated by $\mathcal{T}$ is $O(N^6|\mathcal{P}|^{k+4})$, as results from:

$$O(N^3|\mathcal{P}|^2) \cdot (O(N^3|\mathcal{P}|^{k+2}) + O(N^3) + O(N|\mathcal{P}|^2))$$

While still polynomial, this time complexity is feasible only for small values of N. In general, the time complexity for computing probabilistic features in the full model M_2 increases with both the number of stages in the pipeline and the complexity of the features.

Motivated by efficiency, we decided to use the pipeline model M_3 in which probabilities are computed only over features that appear in the top scoring annotation $\hat{z} = \langle \hat{z}_1, \hat{z}_2 \rangle$, where $\hat{z}_1$ and $\hat{z}_2$ represent the best POS tagging, and the best dependency parse respectively. In order to further speed up the computation of probabilistic features, we made the following approximations:

1. We consider the POS tagging and the dependency parse independent and rewrite $p(\tilde{z}|x)$ as:

$$p(\tilde{z}|x) = p(\tilde{z}_1, \tilde{z}_2|x) \approx p(\tilde{z}_1|x)p(\tilde{z}_2|x)$$

2. We enforce an independence assumption between POS tags. Thus, if $\tilde{z}_1 = \langle t_{i_1} t_{i_2} ... t_{i_k} \rangle$ specifies the tags at k positions in the sentence, then $p(\tilde{z}_1|x)$ is rewritten as:

$$p(t_{i_1} t_{i_2} ... t_{i_k}|x) \approx \prod_{j=1}^{k} p(t_{i_j}|x)$$

3. We also enforce a similar independence assumption between dependency links. Thus, if $\tilde{z}_2 = \langle u_1 \rightarrow v_1 ... u_k \rightarrow v_k \rangle$ specifies k dependency links, then $p(\tilde{z}_2|x)$ is rewritten as:

$$p(u_1 \rightarrow v_1 ... u_k \rightarrow v_k|x) \approx \prod_{l=1}^{k} p(u_l \rightarrow v_l|x)$$

For example, the probability $p(\tilde{z}|x)$ of the feature instance $\textit{NNS}_2 \rightarrow \textit{thought}_4 \leftarrow \textit{RB}_3$ is approximated as:

$$\begin{aligned} p(\tilde{z}|x) &\approx p(\tilde{z}_1|x) \cdot p(\tilde{z}_2|x) \\ p(\tilde{z}_1|x) &\approx p(t_2 = \textit{NNS}|x) \cdot p(t_3 = \textit{RB}|x) \\ p(\tilde{z}_2|x) &\approx p(2 \rightarrow 4|x) \cdot p(3 \rightarrow 4|x) \end{aligned}$$

We will use M'_3 to refer to the resulting model.

4.1 Probabilistic Dependency Features

The probabilistic POS features $p(t_i|x)$ are computed using the forward-backward procedure in CRFs, as described in Section 3.1. To completely specify the pipeline model for NER, we also need an efficient method for computing the probabilistic dependency features $p(u \rightarrow v|x)$, where $u \rightarrow v$ is a dependency edge between positions u and v in the sentence x. MSTPARSER is a large-margin method that computes an unbounded score $s(x, y)$ for any given sentence x and dependency structure $y \in \mathcal{Y}(x)$ using the following edge-based factorization:

$$s(x,y) = \sum_{u\rightarrow v \in y} s(x, u\rightarrow v) = w \sum_{u\rightarrow v \in y} \phi(x, u\rightarrow v)$$

The following three steps describe a general method for associating probabilities with output substructures. The method can be applied whenever a structured output is associated a score value that is unbounded in $\mathbb{R}$, assuming that the score of the entire output structure can be computed efficiently based on a factorization into smaller substructures.

S1. Map the unbounded score $s(x, y)$ from $\mathbb{R}$ into $[0, 1]$ using the *softmax* function (Bishop, 1995):

$$n(x,y) = \frac{e^{s(x,y)}}{\sum_{y \in \mathcal{Y}(x)} e^{s(x,y)}}$$

The normalized score $n(x, y)$ preserves the ranking given by the original score $s(x, y)$. The normalization constant at the denominator can be computed in $O(N^3)$ time by replacing the *max* operator with the *sum* operator inside Eisner's chart parsing algorithm.

S2. Compute a normalized score for the substructure by summing up the normalized scores of all the complete structures that contain it. In our model, dependency edges are substructures, while dependency trees are complete structures. The normalized score will then be computed as:

$$n(x, u\rightarrow v) = \sum_{y \in \mathcal{Y}(x), u\rightarrow v \in y} n(x,y)$$

The sum can be computed in $O(N^3)$ time using a constrained version of the algorithm that computes the normalization constant in step S1. This constrained version of Eisner's algorithm works in a similar manner with the constrained forward backward algorithm by restricting the dependency structures to contain a predefined edge or set of edges.

S3. Use the isotonic regression method of Zadrozny and Elkan (2002) to map the normalized scores $n(x, u\rightarrow v)$ into probabilities $p(u\rightarrow v|x)$. A potential problem with the softmax function is that, depending on the distribution of scores, the exponential transform could dramatically overinflate the higher scores. Isotonic regression, by redistributing the normalized scores inside $[0, 1]$, can alleviate this problem.

4.2 Experimental Results

We test the pipeline model M_3' versus the traditional model M_1 on the task of detecting mentions of person entities in the ACE dataset[2]. We use the standard training – testing split of the ACE 2002 dataset in which the training dataset is also augmented with the documents from the ACE 2003 dataset. The combined dataset contains 674 documents for training and 97 for testing. We implemented the CRF model in MALLET using three different sets of features: **Tree**, **Flat**, and **Full** corresponding to the union of all flat and tree features. The POS tagger and the dependency parser were trained on sections 2-21 of the Penn Treebank, followed by an isotonic regression step on section 23 for the dependency parser. We compute precision recall (PR) graphs by varying a threshold on the token level confidence output by the CRF tagger, and summarize the tagger performance using the area under the curve. Table 4 shows the results obtained by the two models under the three feature settings. The model based on probabilistic features consistently outperforms the traditional model, especially when only tree features are used. Dependency parsing is significantly less accurate than POS tagging. Consequently, the improvement for the tree based model is more substantial than for the flat

Model	Tree	Flat	Full
M_3'	**76.78**	**77.02**	**77.96**
M_1	74.38	76.53	77.02

Table 4: Mention detection results.

[2]URL: http://www.nist.gov/speech/tests/ace

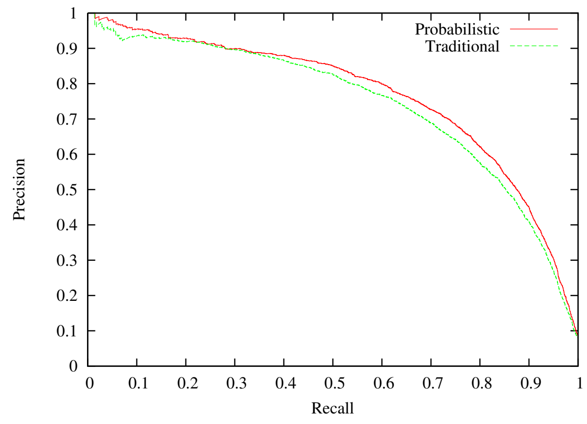


Figure 4: PR graphs for tree features.

model, confirming our expectation that probabilistic features are more useful when upstream stages in the pipeline are less accurate. Figure 4 shows the PR curves obtained for the tree-based models, on which we see a significant 5% improvement in precision over a wide range of recall values.

5 Related Work

In terms of the target task – improving the performance of linguistic pipelines – our research is most related to the work of Finkel et al. (2006). In their approach, output samples are drawn at each stage in the pipeline conditioned on the samples drawn at previous stages, and the final output is determined by a majority vote over the samples from the final stage. The method needs very few samples for tasks such as textual entailment, where the final outcome is binary, in agreement with a theoretical result on the rate of convergence of the voting Gibbs classifier due to Ng and Jordan (2001). While their sampling method is inherently approximate, our full pipeline model M_2 is exact in the sense that feature expectations are computed exactly in polynomial time whenever the inference step at each stage can be done in polynomial time, irrespective of the cardinality of the final output space. Also, the pipeline models M_2 and M_3 and their more efficient alternatives propagate uncertainty during both training and testing through the vector of probabilistic features, whereas the sampling method takes advantage of the probabilistic nature of the outputs only during testing. Overall, the two approaches can be seen as complementary. In order to be applicable with minimal engineering effort, the sampling method needs NLP researchers to write packages that can generate samples from the posterior. Similarly, the new pipeline models could be easily applied in a diverse range of applications, assuming researchers develop packages that can efficiently compute marginals over output substructures.

6 Conclusions and Future Work

We have presented a new, general method for improving the communication between consecutive stages in pipeline models. The method relies on the computation of probabilities for count features, which translates in adding a polynomial factor to the overall time complexity of the pipeline whenever the inference step at each stage is done in polynomial time, which is the case for the vast majority of inference algorithms used in practical NLP applications. We have also shown that additional independence assumptions can make the approach more practical by significantly reducing the time complexity. Existing learning based models can implement the new method by replacing the original feature vector with a more dense vector of probabilistic features[3]. It is essential that every stage in the pipeline produces probabilistic features, and to this end we have described an effective method for associating probabilities with output substructures.

We have shown for NER that simply using the probabilities associated with features that appear only in the top annotation can lead to useful improvements in performance, with minimal engineering effort. In future work we plan to empirically evaluate NER with an approximate version of the full model M_2 which, while more demanding in terms of time complexity, could lead to even more significant gains in accuracy. We also intend to comprehensively evaluate the proposed scheme for computing probabilities by experimenting with alternative normalization functions.

Acknowledgements

We would like to thank Rada Mihalcea and the anonymous reviewers for their insightful comments and suggestions.

[3]The Java source code will be released on my web page.

References

Christopher M. Bishop. 1995. *Neural Networks for Pattern Recogntion*. Oxford University Press.

Aron Culotta and Andrew McCallum. 2004. Confidence estimation for information extraction. In *Proceedings of Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, Boston, MA.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th Conference on Computational linguistics*, pages 340–345, Copenhagen, Denmark.

Jenny R. 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.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of 18th International Conference on Machine Learning (ICML-2001)*, pages 282–289, Williamstown, MA.

M. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL-05)*, pages 91–98, Ann Arbor, Michigan.

Andrew Y. Ng and Michael I. Jordan. 2001. Convergence rates of the Voting Gibbs classifier, with application to bayesian feature selection. In *Proceedings of 18th International Conference on Machine Learning (ICML-2001)*, pages 377–384, Williamstown, MA.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part of speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-96)*, pages 133–141, Philadelphia, PA.

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, pages 1–8, Boston, MA.

Bernhard Schölkopf and Alexander J. Smola. 2002. *Learning with kernels - support vector machines, regularization, optimization and beyond*. MIT Press, Cambridge, MA.

Charles Sutton and Andrew McCallum. 2005. Joint parsing and semantic role labeling. In *CoNLL-05 Shared Task*.

Charles Sutton, Khashayar Rohanimanesh, and Andrew McCallum. 2004. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *Proceedings of 21st International Conference on Machine Learning (ICML-2004)*, pages 783–790, Banff, Canada, July.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. John Wiley & Sons.

Ben Wellner, Andrew McCallum, Fuchun Peng, and Michael Hay. 2004. An integrated, conditional model of information extraction and coreference with application to citation matching. In *Proceedings of 20th Conference on Uncertainty in Artificial Intelligence (UAI-2004)*, Banff, Canada, July.

Bianca Zadrozny and Charles Elkan. 2002. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002)*, Edmonton, Alberta.

Cross-Task Knowledge-Constrained Self Training

Hal Daumé III
School of Computing
University of Utah
Salt Lake City, UT 84112
me@hal3.name

Abstract

We present an algorithmic framework for learning multiple related tasks. Our framework exploits a form of prior knowledge that relates the output spaces of these tasks. We present PAC learning results that analyze the conditions under which such learning is possible. We present results on learning a shallow parser and named-entity recognition system that exploits our framework, showing consistent improvements over baseline methods.

1 Introduction

When two NLP systems are run on the same data, we expect certain constraints to hold between their outputs. This is a form of prior knowledge. We propose a self-training framework that uses such information to significantly boost the performance of one of the systems. The key idea is to perform self-training *only* on outputs that obey the constraints.

Our motivating example in this paper is the task pair: named entity recognition (NER) and shallow parsing (aka syntactic chunking). Consider a hidden sentence with known POS and syntactic structure below. Further consider four potential NER sequences for this sentence.

POS:	NNP	NNP	VBD	TO	NNP	NN
Chunk:	[- NP	-]	[- VP -]	[-PP-]	[- NP -]	[-NP-]
NER1:	[- Per		-]	[- O -]	[-Org-]	[- 0 -]
NER2:	[- Per	-]	[- O -]	[- O -]	[- O -]	[- O -]
NER3:	[- Per	-]	[- O -]	[- O -]	[- Org	-]
NER4:	[- Per	-]	[- O -]	[- O -]	[-Org-]	[- O -]

Without ever seeing the actual sentence, can we guess which NER sequence is correct? NER1 seems wrong because we feel like named entities should not be part of verb phrases. NER2 seems wrong because there is an NNP[1] (proper noun) that is not part of a named entity (word 5). NER3 is amiss because we feel it is unlikely that a *single* name should span more than one NP (last two words). NER4 has none of these problems and seems quite reasonable. In fact, for the hidden sentence, NER4 is correct[2].

The remainder of this paper deals with the problem of formulating such prior knowledge into a workable system. There are similarities between our proposed model and both self-training and co-training; background is given in Section 2. We present a formal model for our approach and perform a simple, yet informative, analysis (Section 3). This analysis allows us to define what good and bad constraints are. Throughout, we use a running example of NER using hidden Markov models to show the efficacy of the method and the relationship between the theory and the implementation. Finally, we present full-blown results on seven different NER data sets (one from CoNLL, six from ACE), comparing our method to several competitive baselines (Section 4). We see that for many of these data sets, less than one hundred labeled NER sentences are required to get state-of-the-art performance, using a discriminative sequence labeling algorithm (Daumé III and Marcu, 2005).

2 Background

Self-training works by learning a model on a small amount of labeled data. This model is then evalu-

[1]When we refer to NNP, we also include NNPS.

[2]The sentence is: “George Bush spoke to Congress today”

ated on a large amount of unlabeled data. Its predictions are assumed to be correct, and it is retrained on the unlabeled data according to its own predictions. Although there is little theoretical support for self-training, it is relatively popular in the natural language processing community. Its success stories range from parsing (McClosky et al., 2006) to machine translation (Ueffing, 2006). In some cases, self-training takes into account *model confidence*.

Co-training (Yarowsky, 1995; Blum and Mitchell, 1998) is related to self-training, in that an algorithm is trained on its own predictions. Where it differs is that co-training learns two *separate* models (which are typically assumed to be independent; for instance by training with disjoint feature sets). These models are both applied to a large repository of unlabeled data. Examples on which these two models *agree* are extracted and treated as labeled for a new round of training. In practice, one often also uses a notion of model confidence and only extracts agreed-upon examples for which both models are confident. The original, and simplest analysis of co-training is due to Blum and Mitchell (1998). It does not take into account confidence (to do so requires a *significantly* more detailed analysis (Dasgupta et al., 2001)), but is useful for understanding the process.

3 Model

We define a formal PAC-style (Valiant, 1994) model that we call the "hints model"[3]. We have an instance space $\mathcal{X}$ and *two* output spaces $\mathcal{Y}_1$ and $\mathcal{Y}_2$. We assume two concept classes $\mathcal{C}_1$ and $\mathcal{C}_2$ for each output space respectively. Let $\mathcal{D}$ be a distribution over $\mathcal{X}$, and $f_1 \in \mathcal{C}_1$ (resp., $f_2 \in \mathcal{C}_2$) be target functions. The goal, of course, is to use a finite sample of examples drawn from $\mathcal{D}$ (and labeled—perhaps with noise—by f_1 and f_2) to "learn" $h_1 \in \mathcal{C}_1$ and $h_2 \in \mathcal{C}_2$, which are good approximations to f_1 and f_2.

So far we have not made use of any notion of constraints. Our expectation is that if we constrain h_1 and h_2 to *agree* (vis-a-vis the example in the Introduction), then we should need fewer labeled examples to learn either. (The agreement should "shrink" the size of the corresponding hypothesis spaces.) To formalize this, let $\chi : \mathcal{Y}_1 \times \mathcal{Y}_2 \to \{0, 1\}$ be a *constraint function.* We say that two outputs $y_1 \in \mathcal{Y}_1$ and $y_2 \in \mathcal{Y}_2$ are *compatible* if $\chi(y_1, y_2) = 1$. We need to assume that χ is correct:

Definition 1. *We say that χ is* correct *with respect to $\mathcal{D}, f_1, f_2$ if whenever x has non-zero probability under $\mathcal{D}$, then $\chi(f_1(x), f_2(x)) = 1$.*

RUNNING EXAMPLE

In our example, $\mathcal{Y}_1$ is the space of all POS/chunk sequences and $\mathcal{Y}_2$ is the space of all NER sequences. We assume that $\mathcal{C}_1$ and $\mathcal{C}_2$ are both represented by HMMs over the appropriate state spaces. The functions we are trying to learn are f_1, the "true" POS/chunk labeler and f_2, the "true" NER labeler. (Note that we assume $f_1 \in \mathcal{C}_1$, which is obviously not true for language.)

Our constraint function χ will require the following for agreement: (1) any NNP must be part of a named entity; (2) any named entity must be a subsequence of a noun phrase. This is precisely the set of constraints discussed in the introduction.

The question is: given this additional source of knowledge (i.e., χ), has the learning problem become easier? That is, can we learn f_2 (and/or f_1) using significantly fewer labeled examples than if we did not have χ? Moreover, we have assumed that χ is *correct*, but is this enough? Intuitively, no: a function χ that returns 1 regardless of its inputs is clearly not useful. Given this, what other constraints must be placed on χ. We address these questions in Sections 3.3. However, first we define our algorithm.

3.1 One-sided Learning with Hints

We begin by considering a simplified version of the "learning with hints" problem. Suppose that all we care about is learning f_2. We have a small amount of data labeled by f_2 (call this D) and a *large* amount of data labeled by f_1 (call this D^{unlab}–"unlab" because as far as f_2 is concerned, it is unlabeled).

RUNNING EXAMPLE

In our example, this means that we have a small amount of labeled NER data and a large amount of labeled POS/chunk data. We use 3500 sentences from CoNLL (Tjong Kim Sang and De Meulder, 2003) as the NER data and section 20-23 of the WSJ (Marcus et al., 1993; Ramshaw and Marcus, 1995) as the POS/chunk data (8936 sentences). We are *only* interested in learning to do NER. Details of the exact HMM setup are in Section 4.2.

[3]The name comes from thinking of our knowledge-based constraints as "hints" to a learner as to what it should do.

We call the following algorithm "One-Sided Learning with Hints," since it aims only to learn f_2:

```
1: Learn $h_2$ directly on $D$
2: For each example $(x, y_1) \in D^{\text{unlab}}$
3:     Compute $y_2 = h_2(x)$
4:     If $\chi(y_1, y_2)$, add $(x, y_2)$ to $D$
5: Relearn $h_2$ on the (augmented) $D$
6: Go to (2) if desired
```

RUNNING EXAMPLE

In step 1, we train an NER HMM on CoNLL. On test data, this model achieves an F-score of 50.8. In step 2, we run this HMM on all the WSJ data, and extract 3145 compatible examples. In step 3, we retrain the HMM; the F-score rises to 58.9.

3.2 Two-sided Learning with Hints

In the two-sided version, we assume that we have a small amount of data labeled by f_1 (call this D_1), a small amount of data labeled by f_2 (call this D_2) and a *large* amount of unlabeled data (call this D^{unlab}). The algorithm we propose for learning hypotheses for both tasks is below:

```
1: Learn $h_1$ on $D_1$ and $h_2$ on $D_2$.
2: For each example $x \in D^{\text{unlab}}$:
3:     Compute $y_1 = h_1(x)$ and $y_2 = h_2(x)$
4:     If $\chi(y_1, y_2)$ add $(x, y_1)$ to $D_1$, $(x, y_2)$ to $D_2$
5: Relearn $h_1$ on $D_1$ and $h_2$ on $D_2$.
6: Go to (2) if desired
```

RUNNING EXAMPLE

We use 3500 examples from NER and 1000 from WSJ. We use the remaining 18447 examples as unlabeled data. The baseline HMMs achieve F-scores of 50.8 and 76.3, respectively. In step 2, we add 7512 examples to each data set. After step 3, the new models achieve F-scores of 54.6 and 79.2, respectively. The gain for NER is lower than before as it is trained against "noisy" syntactic labels.

3.3 Analysis

Our goal is to prove that one-sided learning with hints "works." That is, if C_2 is learnable from large amounts of labeled data, then it is also learnable from small amounts of labeled data and large amounts of f_1-labeled data. This is formalized in Theorem 1 (all proofs are in Appendix A). However, before stating the theorem, we must define an "initial weakly-useful predictor" (terminology from Blum and Mitchell(1998)), and the notion of noisy PAC-learning in the structured domain.

Definition 2. *We say that h is a* weakly-useful predictor *of f if for all y:* $\Pr_{\mathcal{D}}[h(x) = y] \geq \epsilon$ *and* $\Pr_{\mathcal{D}}[f(x) = y \mid h(x) = y' \neq y] \geq \Pr_{\mathcal{D}}[f(x) = y] + \epsilon$.

This definition simply ensures that (1) h is non-trivial: it assigns some non-zero probability to every possible output; and (2) h is somewhat indicative of f. In practice, we use the hypothesis learned on the small amount of training data during step (1) of the algorithm as the weakly useful predictor.

Definition 3. *We say that $\mathcal{C}$ is* PAC-learnable with noise (in the structured setting) *if there exists an algorithm with the following properties. For any $c \in \mathcal{C}$, any distribution $\mathcal{D}$ over $\mathcal{X}$, any $0 \leq \eta \leq 1/|\mathcal{Y}|$, any $0 < \epsilon < 1$, any $0 < \delta < 1$ and any $\eta \leq \eta_0 < 1/|\mathcal{Y}|$, if the algorithm is given access to examples drawn $EX^{\eta}_{SN}(c, \mathcal{D})$ and inputs ϵ, δ and η_0, then with probability at least $1 - \delta$, the algorithm returns a hypothesis $h \in \mathcal{C}$ with error at most ϵ. Here, $EX^{\eta}_{SN}(c, \mathcal{D})$ is a structured noise oracle, which draws examples from $\mathcal{D}$, labels them by c and randomly replaces with another label with prob. η.*

Note here the rather weak notion of noise: entire structures are randomly changed, rather than individual labels. Furthermore, the error is 0/1 loss over the entire structure. Collins (2001) establishes learnability results for the class of hyperplane models under 0/1 loss. While not stated directly in terms of PAC learnability, it is clear that his results apply. Taskar et al. (2005) establish *tighter* bounds for the case of Hamming loss. This suggests that the requirement of 0/1 loss is weaker.

As suggested before, it is not sufficient for χ to simply be *correct* (the constant 1 function is correct, but not useful). We need it to be discriminating, made precise in the following definition.

Definition 4. *We say the* discrimination *of χ for h^0 is* $\Pr_{\mathcal{D}}[\chi(f_1(x), h^0(x))]^{-1}$.

In other words, a constraint function is discriminating when it is unlikely that our weakly-useful predictor h^0 chooses an output that satisfies the constraint. This means that if we *do* find examples (in our unlabeled corpus) that satisfy the constraints, they are likely to be "useful" to learning.

RUNNING EXAMPLE

In the NER HMM, let h^0 be the HMM obtained by training on the small labeled NER data set and f_1 is the true syntactic labels. We approximate $\Pr_{\mathcal{D}}$ by an empirical estimate over the unlabeled distribution. This gives a discrimination is 41.6 for the constraint function defined previously. However, if we compare against "weaker" constraint functions, we see the appropriate trend. The value for the constraint based only on POS tags is 39.1 (worse) and for the NP constraint alone is 27.0 (much worse).

Theorem 1. *Suppose C_2 is PAC-learnable with noise in the structured setting, h_2^0 is a weakly useful predictor of f_2, and χ is correct with respect to $\mathcal{D}, f_1, f_2, h_2^0$, and has discrimination $\geq 2(|\mathcal{Y}| - 1)$. Then C_2 is also PAC-learnable with one-sided hints.*

The way to interpret this theorem is that it tells us that if the initial h_2 we learn in step 1 of the one-sided algorithm is "good enough" (in the sense that it is weakly-useful), then we can use it as specified by the remainder of the one-sided algorithm to obtain an arbitrarily good h_2 (via iterating).

The dependence on $|\mathcal{Y}|$ is the discrimination bound for χ is unpleasant for structured problems. If we wish to find M unlabeled examples that satisfy the hints, we'll need a total of at least $2M(|\mathcal{Y}| - 1)$ total. This dependence can be improved as follows. Suppose that our structure is represented by a graph over vertices V, each of which can take a label from a set Y. Then, $|\mathcal{Y}| = \left|Y^V\right|$, and our result requires that χ be discriminating on an order exponential in V. Under the assumption that χ decomposes over the graph structure (true for our example) and that C_2 is PAC-learnable with per-vertex noise, then the discrimination requirement drops to $2\,|V|\,(|Y| - 1)$.

RUNNING EXAMPLE

In NER, $|Y| = 9$ and $|V| \approx 26$. This means that the values from the previous example look not quite so bad. In the 0/1 loss case, they are compared to 10^{25}; in the Hamming case, they are compared to only 416. The ability of the one-sided algorithm follows the same trends as the discrimination values. Recall the baseline performance is 50.8. With both constraints (and a discrimination value of 41.6), we obtain a score of 58.9. With just the POS constraint (discrimination of 39.1), we obtain a score of 58.1. With just the NP constraint (discrimination of 27.0, we obtain a score of 54.5.

The final question is how one-sided learning relates to two-sided learning. The following definition and easy corollary shows that they are related in the obvious manner, but depends on a notion of uncorrelation between h_1^0 and h_2^0.

Definition 5. *We say that h_1 and h_2 are* uncorrelated *if* $\Pr_{\mathcal{D}}\left[h_1(x) = y_1 \mid h_2(x) = y_2, x\right] = \Pr_{\mathcal{D}}\left[h_1(x) = y_1 \mid x\right]$.

Corollary 1. *Suppose C_1 and C_2 are both PAC-learnable in the structured setting, h_1^0 and h_2^0 are weakly useful predictors of f_1 and f_2, and χ is correct with respect to $\mathcal{D}, f_1, f_2, h_1^0$ and h_2^0, and has discrimination $\geq 4(|\mathcal{Y}| - 1)^2$ (for 0/1 loss) or $\geq 4\,|V|^2\,(|Y| - 1)^2$ (for Hamming loss), and that h_1^0 and h_2^0 are uncorrelated. Then C_1 and C_2 are also PAC-learnable with two-sided hints.*

Unfortunately, Corollary 1 depends *quadratically* on the discrimination term, unlike Theorem 1.

4 Experiments

In this section, we describe our experimental results. We have already discussed some of them in the context of the running example. In Section 4.1, we briefly describe the data sets we use. A full description of the HMM implementation and its results are in Section 4.2. Finally, in Section 4.3, we present results based on a competitive, discriminatively-learned sequence labeling algorithm. All results for NER and chunking are in terms of F-score; all results for POS tagging are accuracy.

4.1 Data Sets

Our results are based on syntactic data drawn from the Penn Treebank (Marcus et al., 1993), specifically the portion used by CoNLL 2000 shared task (Tjong Kim Sang and Buchholz, 2000). Our NER data is from two sources. The first source is the CoNLL 2003 shared task date (Tjong Kim Sang and De Meulder, 2003) and the second source is the 2004 NIST Automatic Content Extraction (Weischedel, 2004). The ACE data constitute six separate data sets from six domains: weblogs (wl), newswire (nw), broadcast conversations (bc), United Nations (un), direct telephone speech (dts) and broadcast news (bn). Of these, bc, dts and bn are all speech data sets. All the examples from the previous sections have been limited to the CoNLL data.

4.2 HMM Results

The experiments discussed in the preceding sections are based on a generative hidden Markov model for both the NER and syntactic chunking/POS tagging tasks. The HMMs constructed use first-order transitions and emissions. The emission vocabulary is pruned so that any word that appears ≤ 1 time in the training data is replaced by a unique `*unknown*` token. The transition and emission probabilities are smoothed with Dirichlet smoothing, $\alpha = 0.001$ (this was not-aggressively tuned by hand on one setting). The HMMs are implemented as finite state models in the Carmel toolkit (Graehl and Knight, 2002).

The various compatibility functions are also implemented as finite state models. We implement them as a transducer *from* POS/chunk labels *to* NER labels (though through the reverse operation, they can obviously be run in the opposite direction). The construction is with a single state with transitions:

- (NNP,?) maps to B-* and I-*
- (?,B-NP) maps to B-* and O
- (?,I-NP) maps to B-*, I-* and O
- Single exception: (NNP,x), where x is *not* an NP tag maps to anything (this is simply to avoid empty composition problems). This occurs in 100 of the $212k$ words in the Treebank data and more rarely in the automatically tagged data.

4.3 One-sided Discriminative Learning

In this section, we describe the results of one-sided discriminative labeling with hints. We use the true syntactic labels from the Penn Treebank to derive the constraints (this is roughly 9000 sentences). We use the LaSO sequence labeling software (Daumé III and Marcu, 2005), with its built-in feature set.

Our goal is to analyze two things: (1) what is the effect of the amount of labeled NER data? (2) what is the effect of the amount of labeled syntactic data from which the hints are constructed?

To answer the first question, we keep the amount of syntactic data fixed (at 8936 sentences) and vary the amount of NER data in $N \in \{100, 200, 400, 800, 1600\}$. We compare models with and without the default gazetteer information from the LaSO software. We have the following models for comparison:

- A default "Baseline" in which we simply train the NER model without using syntax.

	Hints vs **Base**	Self-T vs **Base**	Hints vs **Self-T**
Win	29	20	24
Tie	6	12	11
Lose	0	3	0

Table 1: Comparison between hints, self-training and the (best) baseline for varying amount of labeled data.

- In "POS-feature", we do the same thing, but we first label the NER data using a tagger/chunker trained on the 8936 syntactic sentences. These labels are used as features for the baseline.
- A "Self-training" setting where we use the 8936 syntactic sentences as "unlabeled," label them with our model, and then train on the results. (This is equivalent to a hints model where $\chi(\cdot,\cdot) = 1$ is the constant 1 function.) We use model confidence as in Blum and Mitchell (1998).[4]

The results are shown in Figure 1. The trends we see are the following:

- More data always helps.
- Self-training usually helps over the baseline (though not always: for instance in wl and parts of cts and bn).
- Adding the gazetteers help.
- Adding the syntactic features helps.
- Learning with hints, especially for $\leq$ 1000 training data points, helps significantly, even over self-training.

We further compare the algorithms by looking at how many training setting has each as the winner. In particular, we compare both hints and self-training to the two baselines, and then compare hints to self-training. If results are not significant at the 95% level (according to McNemar's test), we call it a tie. The results are in Table 1.

In our second set of experiments, we consider the role of the syntactic data. For this experiment, we hold the number of NER labeled sentences constant (at $N = 200$) and vary the amount of syntactic data in $M \in \{500, 1000, 2000, 4000, 8936\}$. The results of these experiments are in Figure 2. The trends are:

- The POS feature is relatively insensitive to the amount of syntactic data—this is most likely because it's weight is discriminatively adjusted

[4]Results without confidence were significantly worse.

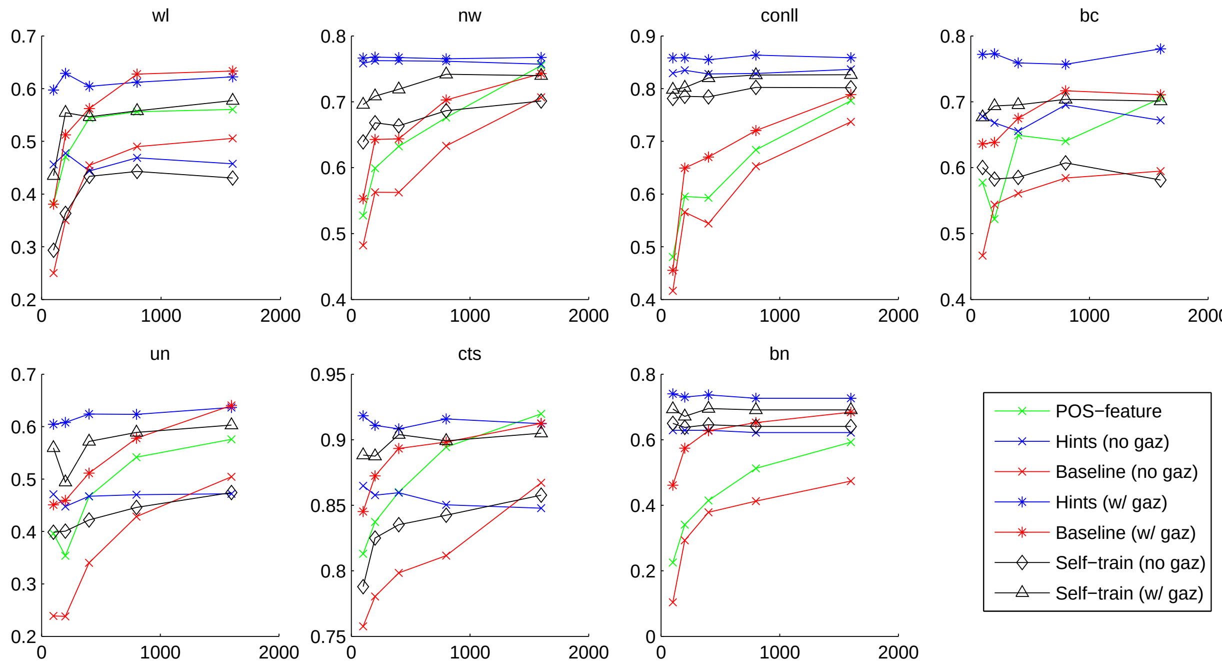


Figure 1: Results of varying the amount of NER labeled data, for a fixed amount ($M = 8936$) of syntactic data.

	Hints vs **Base**	**Self-T** vs **Base**	**Hints** vs **Self-T**
Win	34	28	15
Tie	1	7	20
Lose	0	0	0

Table 2: Comparison between hints, self-training and the (best) baseline for varying amount of unlabeled data.

by LaSO so that if the syntactic information is bad, it is relatively ignored.

- Self-training performance often *degrades* as the amount of syntactic data increases.
- The performance of learning with hints increases steadily with more syntactic data.

As before, we compare performance between the different models, declaring a "tie" if the difference is not statistically significant at the 95% level. The results are in Table 2.

In experiments not reported here to save space, we experimented with several additional settings. In one, we weight the unlabeled data in various ways: (1) to make it equal-weight to the labeled data; (2) at 10% weight; (3) according to the score produced by the first round of labeling. None of these had a significant impact on scores; in a few cases performance went up by $\ll 1$, in a few cases, performance went down about the same amount.

4.4 Two-sided Discriminative Learning

In this section, we explore the use of two-sided discriminative learning to boost the performance of our syntactic chunking, part of speech tagging, and named-entity recognition software. We continue to use LaSO (Daumé III and Marcu, 2005) as the sequence labeling technique.

The results we present are based on attempting to improve the performance of a state-of-the-art system train on *all* of the training data. (This is in contrast to the results in Section 4.3, in which the effect of using limited amounts of data was explored.) For the POS tagging and syntactic chunking, we being with all 8936 sentences of training data from CoNLL. For the named entity recognition, we limit our presentation to results from the CoNLL 2003 NER shared task. For this data, we have roughly $14k$ sentences of training data, all of which are used. In both cases, we reserve 10% as development data. The development data is use to do early stopping in LaSO.

As unlabeled data, we use $1m$ sentences extracted from the North American National Corpus of En-

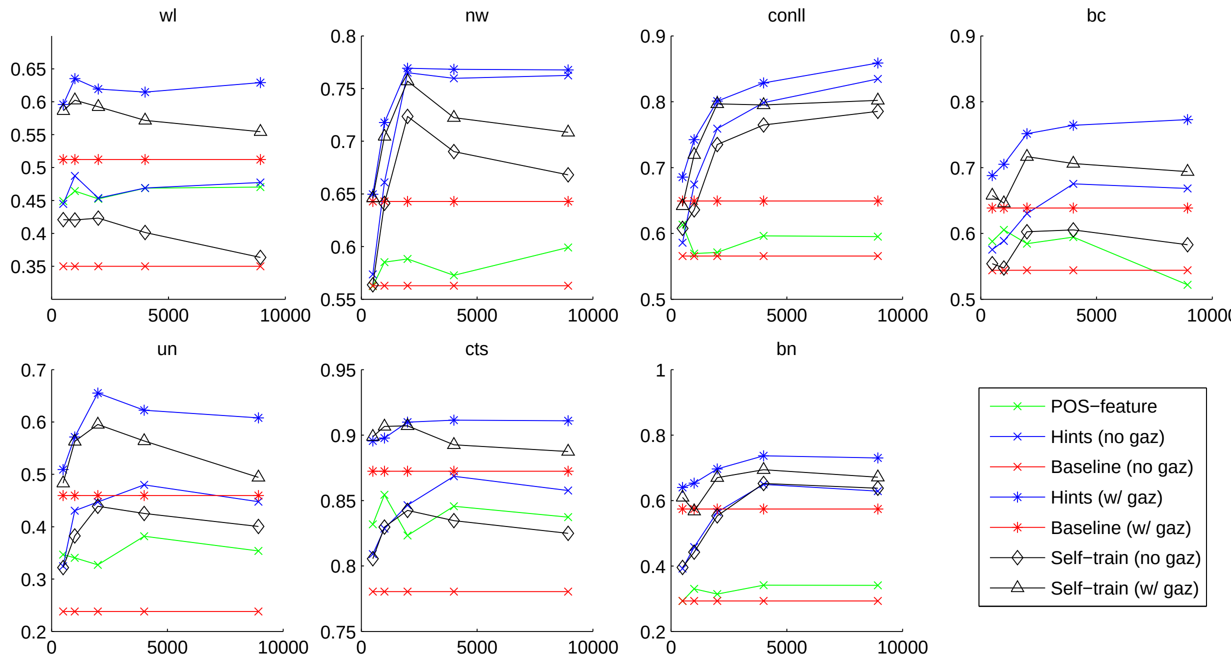


Figure 2: Results of varying amount of syntactic data for a fixed amount of NER data ($N = 200$ sentences).

glish (previously used for self-training of parsers (McClosky et al., 2006)). These $1m$ sentences were selected by dev-set relativization against the union of the two development data sets.

Following similar ideas to those presented by Blum and Mitchell (1998), we employ two slight modifications to the algorithm presented in Section 3.2. First, in step (2b) instead of adding *all* allowable instances to the labeled data set, we only add the top R (for some hyper-parameter R), where "top" is determined by average model confidence for the two tasks. Second, Instead of using the *full* unlabeled set to label at each iteration, we begin with a random subset of $10R$ unlabeled examples and another add random $10R$ every iteration.

We use the same baseline systems as in one-sided learning: a Baseline that learns the two tasks independently; a variant of the Baseline on which the output of the POS/chunker is used as a feature for the NER; a variant based on self-training; the hints-based method. In all cases, we *do* use gazetteers. We run the hints-based model for 10 iterations. For self-training, we use $10R$ unlabeled examples (so that it had access to the same amount of unlabeled data as the hints-based learning after all 10 iterations). We used three values of R: 50, 100, 500. We select the

	Chunking	**NER**
Baseline	94.2	87.5
w/POS	N/A	88.0
Self-train		
$R = 50$	94.2	88.0
$R = 100$	94.3	88.6
$R = 500$	94.1	88.4
Hints		
$R = 50$	94.2	88.5
$R = 100$	94.3	89.1
$R = 500$	94.3	89.0

Table 3: Results on two-sided learning with hints.

best-performing model (by the dev data) over these ten iterations. The results are in Table 3.

As we can see, performance for syntactic chunking is relatively stagnant: there are no significant improvements for any of the methods over the baseline. This is not surprising: the form of the constraint function we use tells us a *lot* about the NER task, but relatively little about the syntactic chunking task. In particular, it tells us nothing about phrases other than NPs. On the other hand, for NER, we see that both self-training and learning with hints improve over the baseline. The improvements are not enormous, but *are* significant (at the 95% level, as measured by McNemar's test). Unfortunately, the improvements for learning with hints over the self-training model are only significant at the 90% level.

5 Discussion

We have presented a method for simultaneously learning two tasks using prior knowledge about the relationship between their outputs. This is related to *joint inference* (Daumé III et al., 2006). However, we do not require that that a single data set be labeled for multiple tasks. In all our examples, we use separate data sets for shallow parsing as for named-entity recognition. Although all our experiments used the LaSO framework for sequence labeling, there is *noting* in our method that assumes any particular learner; alternatives include: conditional random fields (Lafferty et al., 2001), independent predictors (Punyakanok and Roth, 2001), max-margin Markov networks (Taskar et al., 2005), etc.

Our approach, both algorithmically and theoretically, is most related to ideas in co-training (Blum and Mitchell, 1998). The key difference is that in co-training, one assumes that the two "views" are on the *inputs*; here, we can think of the two output spaces as being the difference "views" and the compatibility function χ being a method for reconciling these two views. Like the pioneering work of Blum and Mitchell, the algorithm we employ in practice makes use of incrementally augmenting the unlabeled data and using model confidence. Also like that work, we do not currently have a theoretical framework for this (more complex) model.[5] It would also be interesting to explore *soft* hints, where the range of χ is $[0, 1]$ rather than $\{0, 1\}$.

Recently, Ganchev et al. (2008) proposed a co-regularization framework for learning across multiple related tasks with different output spaces. Their approach hinges on a constrained EM framework and addresses a quite similar problem to that addressed by this paper. Chang et al. (2008) also propose a "semisupervised" learning approach quite similar to our own model. The show very promising results in the context of semantic role labeling. Given the apparent (very!) recent interest in this problem, it would be ideal to directly compare the different approaches.

In addition to an analysis of the theoretical properties of the algorithm presented, the most compelling avenue for future work is to apply this framework to other task pairs. With a little thought, one can imagine formulating compatibility functions between tasks like discourse parsing and summarization (Marcu, 2000), parsing and word alignment, or summarization and information extraction.

Acknowledgments

Many thanks to three anonymous reviewers of this papers whose suggestions greatly improved the work and the presentation. This work was partially funded by NSF grant IIS 0712764.

A Proofs

The proof of Theorem 1 closes follows that of Blum and Mitchell (1998).

Proof (Theorem 1, sketch). Use the following notation: $c_k = \Pr_{\mathcal{D}}[h(x) = k]$, $p_l = \Pr_{\mathcal{D}}[f(x) = l]$, $q_{l|k} = \Pr_{\mathcal{D}}[f(x) = l \mid h(x) = k]$. Denote by $\mathcal{A}$ the set of outputs that satisfy the constraints. We are interested in the probability that $h(x)$ is erroneous, given that $h(x)$ satisfies the constraints:

$$
\begin{aligned}
&p\left(h(x) \in \mathcal{A}\backslash\{l\} \mid f(x) = l\right) \\
&= \sum_{k \in \mathcal{A}\backslash\{l\}} p\left(h(x) = k \mid f(x) = l\right) = \sum_{k \in \mathcal{A}\backslash\{l\}} c_k q_{l|k}/p_l \\
&\le \sum_{k \in \mathcal{A}} c_k (|\mathcal{Y}| - 1 + \epsilon \sum_{l \ne k} 1/p_l) \le 2 \sum_{k \in \mathcal{A}} c_k (|\mathcal{Y}| - 1)
\end{aligned}
$$

Here, the second step is Bayes' rule plus definitions, the third step is by the weak initial hypothesis assumption, and the last step is by algebra. Thus, in order to get a probability of error at most η, we need $\sum_{k \in \mathcal{A}} c_k = \Pr[h(x) \in \mathcal{A}] \le \eta/(2(|\mathcal{Y}| - 1))$. □

The proof of Corollary 1 is straightforward.

Proof (Corollary 1, sketch). Write out the probability of error as a double sum over true labels y_1, y_2 and predicted labels $\hat{y}_1, \hat{y}_2$ subject to $\chi(\hat{y}_1, \hat{y}_2)$. Use the uncorrelation assumption and Bayes' to split this into the product two terms as in the proof of Theorem 1. Bound as before. □

[5]Dasgupta et al. (2001) proved, three years later, that a formal model roughly equivalent to the actual Blum and Mitchell algorithm *does* have solid theoretical foundations.

References

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the Conference on Computational Learning Theory (COLT)*, pages 92–100.

Ming-Wei Chang, Lev Ratinov, Nicholas Rizzolo, and Dan Roth. 2008. Learning and inference with constraints. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Michael Collins. 2001. Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods. In *International Workshop on Parsing Technologies (IWPT)*.

Sanjoy Dasgupta, Michael Littman, and David McAllester. 2001. PAC generalization bounds for co-training. In *Advances in Neural Information Processing Systems (NIPS)*.

Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Hal Daumé III, Andrew McCallum, Ryan McDonald, Fernando Pereira, and Charles Sutton, editors. 2006. *Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*. Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics and Human Language Technology (NAACL/HLT).

Kuzman Ganchev, Joao Graca, John Blitzer, and Ben Taskar. 2008. Multi-view learning over structured and non-identical outputs. In *Proceedings of the Converence on Uncertainty in Artificial Intelligence (UAI)*.

Jonathan Graehl and Kevin Knight. 2002. Carmel finite state transducer package. `http://www.isi.edu/licensed-sw/carmel/`.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Daniel Marcu. 2000. *The Theory and Practice of Discourse Parsing and Summarization*. The MIT Press, Cambridge, Massachusetts.

Mitch Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics and Human Language Technology (NAACL/HLT)*.

Vasin Punyakanok and Dan Roth. 2001. The use of classifiers in sequential inference. In *Advances in Neural Information Processing Systems (NIPS)*.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora*.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. 2005. Learning structured prediction models: A large margin approach. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 897–904.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*.

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

Nicola Ueffing. 2006. Self-training for machine translation. In *NIPS workshop on Machine Learning for Multilingual Information Access*.

Leslie G. Valiant. 1994. A theory of the learnable. *Annual ACM Symposium on Theory of Computing*, pages 436–445.

Ralph Weischedel, editor. 2004. *Automatic Content Extraction Workshop (ACE-2004)*, Alexandria, Virginia, September 20–22.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.

Online Methods for Multi-Domain Learning and Adaptation

Mark Dredze and **Koby Crammer**
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104 USA
{mdredze,crammer}@cis.upenn.edu

Abstract

NLP tasks are often domain specific, yet systems can learn behaviors across multiple domains. We develop a new multi-domain online learning framework based on parameter combination from multiple classifiers. Our algorithms draw from multi-task learning and domain adaptation to adapt multiple source domain classifiers to a new target domain, learn across multiple similar domains, and learn across a large number of disparate domains. We evaluate our algorithms on two popular NLP domain adaptation tasks: sentiment classification and spam filtering.

1 Introduction

Statistical classifiers routinely process millions of websites, emails, blogs and other text every day. Variability across different data sources means that training a single classifier obscures differences and separate classifiers ignore similarities. Similarly, adding new domains to existing systems requires adapting existing classifiers.

We present new online algorithms for three *multi-domain learning* scenarios: adapting existing classifiers to new domains, learning across multiple similar domains and scaling systems to many disparate domains. Multi-domain learning combines characteristics of both multi-task learning and domain adaptation and drawing from both areas, we develop a multi-classifier parameter combination technique for confidence-weighted (CW) linear classifiers (Dredze et al., 2008). We focus on online algorithms that scale to large amounts of data.

Next, we describe multi-domain learning and review the CW algorithm. We then consider our three settings using multi-classifier parameter combination. We conclude with related work.

2 Multi-Domain Learning

In online multi-domain learning, each instance $\boldsymbol{x}$ is drawn from a domain d specific distribution $\boldsymbol{x} \sim \mathcal{D}_d$ over a vectors space $\mathbb{R}^N$ and labeled with a domain specific function f_d with label $y \in \{-1, +1\}$ (for binary classification.) On round i the classifier receives instance $\boldsymbol{x}_i$ and domain identifier d_i and predicts label $\hat{y}_i \in \{-1, +1\}$. It then receives the true label $y_i \in \{-1, +1\}$ and updates its prediction rule.

As an example, consider a multi-user spam filter, which must give high quality predictions for new users (without new user data), learn on multiple users simultaneously and scale to thousands of accounts. While a single classifier trained on all users would generalize across users and extend to new users, it would fail to learn user-specific preferences. Alternatively, separate classifiers would capture user-specific behaviors but would not generalize across users. The approach we take to solving multi-domain problems is to combine domain-specific classifiers. In the adaptation setting, we combine source domain classifiers for a new target domain. For learning across domains, we combine domain-specific classifiers and a shared classifier learned across all domains. For learning across disparate domains we learn which domain-specific and shared classifiers to combine.

Multi-domain learning combines properties of both multi-task learning and domain adaptation. As in multi-task learning, we consider domains that are labeled with different classification functions. For example, one user may enjoy some emails that another user considers spam: differing in their classification function. The goal of multi-task learning is to generalize across tasks/domains (Dekel et al., 2006; Evgeniou and Pontil, 2004). Furthermore, as in domain adaptation, some examples are draw from different distributions. For example, one user may receive emails about engineering while another about art, differing in their distribution over features. Domain adaptation deals with these feature distribution changes (Blitzer et al., 2007; Jiang and Zhai, 2007). Our work combines these two areas by learning both across distributions and behaviors or functions.

3 Confidence-Weighted Linear Classifiers

Confidence-weighted (CW) linear classification (Dredze et al., 2008), a new online algorithm, maintains a probabilistic measure of parameter confidence, which may be useful in combining parameters from different domain distributions. We summarize CW learning to familiarize the reader.

Parameter confidence is formalized by a Gaussian distribution over weight vectors with mean $\boldsymbol{\mu} \in \mathbb{R}^N$ and diagonal covariance $\Sigma \in \mathbb{R}^{N \times N}$. The values μ_j and $\Sigma_{j,j}$ represent knowledge of and confidence in the parameter for feature j. The smaller $\Sigma_{j,j}$, the more confidence we have in the mean parameter value μ_j. In this work we consider diagonal covariance matrices to scale to NLP data.

A model predicts the highest probability label,

$$\arg \max_{y \in \{\pm 1\}} \Pr_{\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)} \left[y_i (\boldsymbol{w} \cdot \boldsymbol{x}_i) \geq 0 \right] \ .$$

The Gaussian distribution over parameter vectors $\boldsymbol{w}$ induces a univariate Gaussian distribution over the score $S_i = \boldsymbol{w} \cdot \boldsymbol{x}_i$ parameterized by $\boldsymbol{\mu}$, Σ and the instance $\boldsymbol{x}_i$: $S_i \sim \mathcal{N}\left(\mu_i, \sigma_i^2\right)$, with mean $\mu_i = \boldsymbol{\mu} \cdot \boldsymbol{x}_i$ and variance $\sigma_i^2 = \boldsymbol{x}_i^\top \Sigma \boldsymbol{x}_i$.

The CW algorithm is inspired by the Passive Aggressive (PA) update (Crammer et al., 2006) — which ensures a positive margin while minimizing parameter change. CW replaces the Euclidean distance used in the PA update with the Kullback-Leibler (KL) divergence over Gaussian distributions. It also replaces the minimal margin constraint with a minimal probability constraint: with some given probability $\eta \in (0.5, 1]$ a drawn classifier will assign the correct label. This strategy yields the following objective solved on each round of learning:

$$\begin{aligned} &\min \mathbb{D}_{\mathrm{KL}} \left(\mathcal{N}\left(\boldsymbol{\mu}, \Sigma\right) \parallel \mathcal{N}\left(\boldsymbol{\mu}_i, \Sigma_i\right) \right) \\ &\text{s.t. } \Pr\left[y_i \left(\boldsymbol{w} \cdot \boldsymbol{x}_i \right) \geq 0 \right] \geq \eta \ , \end{aligned}$$

where $(\boldsymbol{\mu}_i, \Sigma_i)$ are the parameters on round i and $\boldsymbol{w} \sim \mathcal{N}\left(\boldsymbol{\mu}, \Sigma\right)$. The constraint ensures that the resulting parameters $\left(\boldsymbol{\mu}_{i+1}, \Sigma_{i+1}\right)$ will correctly classify $\boldsymbol{x}_i$ with probability at least η. For convenience we write $\phi = \Phi^{-1}(\eta)$, where Φ is the cumulative function of the normal distribution. The optimization problem above is not convex, but a closed form approximation of its solution has the following additive form: $\boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_i + \alpha_i y_i \Sigma_i \boldsymbol{x}_i$ and $\Sigma_{i+1}^{-1} = \Sigma_i^{-1} + 2\alpha_i \phi \boldsymbol{x}_i \boldsymbol{x}_i^\top$ for,

$$\alpha_i = \frac{-(1+2\phi\mu_i) + \sqrt{(1+2\phi\mu_i)^2 - 8\phi\left(\mu_i - \phi\sigma_i^2\right)}}{4\phi\sigma_i^2} \ .$$

Each update changes the feature weights $\boldsymbol{\mu}$, and increases confidence (variance Σ always decreases). We employ CW classifiers since they provide confidence estimates, which are useful for classifier combination. Additionally, since we require per-parameter confidence estimates, other confidence based classifiers are not suitable for this setting.

4 Multi-Classifier Parameter Combination

The basis of our approach to multi-domain learning is to combine the parameters of CW classifiers from separate domains while respecting parameter confidence. A combination method takes M CW classifiers each parameterized by its own mean and variance parameters $\{(\boldsymbol{\mu}^m, \Sigma^m)\}_{m=1}^M$ and produces a single combined classifier $(\boldsymbol{\mu}^c, \Sigma^c)$. A simple technique would be to average the parameters of classifiers into a new classifier. However, this ignores the difference in feature distributions. Consider for example that the weight associated with some word in a source classifier has a value of 0. This could either mean that the word is very rare or that it is neutral for prediction (like the work "the"). The information captured by the variance parameter allow us to distinguish between the two cases: an high-variance indicates a lack of confidence in the value of the weight vectors because of small number of examples (first case), and vise-versa, small-variance indicates that the value of the weight is based on plenty of evidence. We favor combinations sensitive to this distinction.

Since CW classifiers are Gaussian distributions, we formalize classifier parameter combination as finding a new distribution that minimizes the weighted-divergence to a set of given distributions:

$$(\boldsymbol{\mu}^c, \Sigma^c) = \arg\min \sum_m^M \mathbb{D}((\boldsymbol{\mu}^c, \Sigma^c) || (\boldsymbol{\mu}^m, \Sigma^m) \, ; \, \mathbf{b}^m) \, ,$$

where (since Σ is diagonal),

$$\mathbb{D}((\boldsymbol{\mu}^c, \Sigma^c) || (\boldsymbol{\mu}, \Sigma) \, ; \, \mathbf{b}) = \sum_f^N b_f D((\boldsymbol{\mu}^c_f, \Sigma^c_{f,f}) || (\boldsymbol{\mu}_f, \Sigma_{f,f})) \, .$$

The (classifier specific) importance-weights $\mathbf{b}^m \in \mathbb{R}^N_+$ are used to weigh certain parameters of some domains differently in the combination. When $\mathbb{D}$ is the Euclidean distance (L2), we have,

$$D((\boldsymbol{\mu}^c_f, \Sigma^c_{f,f}) || (\boldsymbol{\mu}_f, \Sigma_{f,f})) = (\boldsymbol{\mu}^c_f - \boldsymbol{\mu}_f)^2 + (\Sigma^c_{f,f} - \Sigma_{f,f})^2 \, .$$

and we obtain:

$$\boldsymbol{\mu}^c_f = \frac{1}{\sum_m^M b^m_f} \sum_m^M b^m_f \boldsymbol{\mu}^m_f ,$$

$$\Sigma^c_{f,f} = \frac{1}{\sum_{m \in M} b^m_f} \sum_m^M b^m_f \Sigma^m_{f,f} \, . \qquad (1)$$

Note that this is a (weighted) average of parameters. The other case we consider is when $\mathbb{D}$ is a weighted KL divergence we obtain a weighting of $\boldsymbol{\mu}$ by Σ^{-1}:

$$\boldsymbol{\mu}^c_f = \left(\sum_m^M (\Sigma^m_{f,f})^{-1} b^m_f \right)^{-1} \sum_m^M (\Sigma^m_{f,f})^{-1} \boldsymbol{\mu}^m_f b^m_f$$

$$(\Sigma^c)^{-1} = \left(M \sum_m^M b^m_f \right)^{-1} \sum_m^M (\Sigma^m_f)^{-1} {b_f}^m \, . \qquad (2)$$

While each parameter is weighed by its variance in the KL, we can also explicitly encode this behavior as $b^m_f = a - \Sigma^m_{f,f} \geq 0$, where a is the initialization value for $\Sigma^m_{f,f}$. We call this weighting "variance" as opposed to a uniform weighting of parameters ($b^m_f = 1$). We therefore have two combination methods (L2 and KL) and two weighting methods (uniform and variance).

5 Datasets

For evaluation we selected two domain adaptation datasets: spam (Jiang and Zhai, 2007) and sentiment (Blitzer et al., 2007). The spam data contains two tasks, one with three users (task A) and one with 15 (task B). The goal is to classify an email (bag-of-words) as either spam or ham (not-spam) and each user may have slightly different preferences and features. We used 700 and 100 training messages for each user for task A and B respectively and 300 test emails for each user.

The sentiment data contains product reviews from Amazon for four product types: books, dvds, electronics and kitchen appliances and we extended this with three additional domains: apparel, music and videos. We follow Blitzer *et. al.* for feature extraction. We created different datasets by modifying the decision boundary using the ordinal rating of each instance (1-5 stars) and excluding boundary instances. We use four versions of this data:

- **All** - 7 domains, one per product type
- **Books** - 3 domains of books with the binary decision boundary set to 2, 3 and 4 stars
- **DVDs** - Same as *Books* but with DVD reviews
- **Books+DVDs** - Combined *Books* and *DVDs*

The *All* dataset captures the typical domain adaptation scenario, where each domain has the same decision function but different features. *Books* and *DVDs* have the opposite problem: the same features but different classification boundaries. *Books+DVDs* combines both issues. Experiments use 1500 training and 100 test instances per domain.

6 Multi-Domain Adaptation

We begin by examining the typical domain adaptation scenario, but from an online perspective since learning systems often must adapt to new users or domains quickly and with no training data. For example, a spam filter with separate classifiers trained on each user must also classify mail for a new user. Since other user's training data may have been deleted or be private, the existing classifiers must be combined for the new user.

		Train				L2		KL	
Target Domain		*All Src*	*Target*	*Best Src*	*Avg Src*	*Uniform*	*Variance*	*Uniform*	*Variance*
Spam	user0	3.85	1.80	4.80	8.26	5.25	4.63	4.53	**4.32**
	user1	3.57	3.17	4.28	6.91	4.53	**3.80**	4.23	3.83
	user2	3.30	2.40	3.77	5.75	4.75	**4.60**	4.93	4.67
Sentiment	apparel	12.32	12.02	14.12	21.15	14.03	**13.18**	13.50	13.48
	books	16.85	18.95	22.95	25.76	19.58	**18.63**	19.53	19.05
	dvd	13.65	17.40	17.30	21.89	15.53	**13.73**	14.48	14.15
	kitchen	13.65	14.40	15.52	22.88	16.68	15.10	14.78	**14.02**
	electronics	15.00	14.93	15.52	23.84	18.75	17.37	17.45	**16.82**
	music	18.20	18.30	20.75	24.19	18.38	**17.83**	18.10	18.22
	video	17.00	19.27	19.43	25.78	17.13	**16.25**	16.33	16.42

Table 1: Test error for multi-source adaptation on sentiment and spam data. Combining classifiers improves over selecting a single classifier a priori (*Avg Src*).

We combine the existing user-specific classifiers into a single new classifier for a new user. Since nothing is known about the new user (their decision function), each source classifier may be useful. However, feature similarity – possibly measured using unlabeled data – could be used to weigh source domains. Specifically, we combine the parameters of each classifier according to their confidence using the combination methods described above.

We evaluated the four combination strategies – L2 vs. KL, uniform vs. variance – on spam and sentiment data. For each evaluation, a single domain was held out for testing while separate classifiers were trained on each source domain, i.e. no target training. Source classifiers are then combined and the combined classifier is evaluated on the test data (400 instances) of the target domain. Each classifier was trained for 5 iterations over the training data (to ensure convergence) and each experiment was repeated using 10-fold cross validation. The CW parameter ϕ was tuned on a single randomized run for each experiment. We include several baselines: training on target data to obtain an upper bound on performance (*Target*), training on all source domains together, a useful strategy if all source data is maintained (*All Src*), selecting (with omniscience) the best performing source classifier on target data (*Best Src*), and the expected real world performance of randomly selecting a source classifier (*Avg Src*).

While at least one source classifier achieved high performance on the target domain (*Best Src*), the correct source classifier cannot be selected without target data and selecting a random source classifier yields high error. In contrast, a combined classifier almost always improved over the best source domain classifier (table 1). That some of our results improve over the best training scenario is likely caused by increased training data from using multiple domains. Increases over all available training data are very interesting and may be due to a regularization effect of training separate models.

The L2 methods performed best and KL improved 7 out of 10 combinations. Classifier parameter combination can clearly yield good classifiers without prior knowledge of the target domain.

7 Learning Across Domains

In addition to adapting to new domains, multi-domain systems should learn common behaviors across domains. Naively, we can assume that the domains are either sufficiently similar to warrant one classifier or different enough for separate classifiers. The reality is often more complex. Instead, we maintain shared and domain-specific parameters and combine them for learning and prediction.

Multi-task learning aims to learn common behaviors across related problems, a similar goal to multi-domain learning. The primary difference is the nature of the domains/tasks: in our setting each domain is the same task but differs in the types of features in addition to the decision function. A multi-task approach can be adapted to our setting by using our classifier combination techniques.

	Spam		*Sentiment*			
Method	*Task A*	*Task B*	*Books*	*DVD*	*Books+DVD*	*All*
Single	3.88	8.75	23.7	25.11	23.26	16.57
Separate	5.46	14.53	22.22	21.64	21.23	21.89
Feature Splitting	4.16	8.93	15.65	16.20	14.60	17.45
MDR	4.09	9.18	15.65	15.12	13.76	17.45
MDR+L2	4.27	8.61	**12.70**	14.95	12.73	**17.16**
MDR+L2-Var	**3.75**	**7.52**	12.90	14.21	**12.52**	17.37
MDR+KL	4.32	9.22	13.51	**13.81**	13.32	17.20
MDR+KL-Var	4.02	8.70	14.93	14.03	14.22	18.40

Table 2: Online training error for learning across domains.

	Spam		*Sentiment*			
Method	*Task A*	*Task B*	*Books*	*DVD*	*Books+DVD*	*All*
Single	2.11	5.60	18.43	18.67	19.08	14.09
Separate	2.43	8.5	18.87	15.97	16.45	17.23
Feature Splitting	1.94	5.51	9.97	9.70	9.05	14.73
MDR	1.94	5.69	9.97	8.33	8.20	14.73
MDR+L2	**1.87**	5.16	6.63	7.97	7.62	**14.20**
MDR+L2-Var	1.90	**4.78**	**6.40**	7.83	**7.30**	14.33
MDR+KL	1.94	5.61	8.37	**7.07**	8.43	14.60
MDR+KL-Var	1.97	5.46	9.40	7.50	8.05	15.50

Table 3: Test data error: learning across domains (MDR) improves over the baselines and Daumé (2007).

We seek to learn domain specific parameters guided by shared parameters. Dekel et al. (2006) followed this approach for an online multi-task algorithm, although they did not have shared parameters and assumed that a training round comprised an example from each task. Evgeniou and Pontil (2004) achieved a similar goal by using shared parameters for multi-task regularization. Specifically, they assumed that the weight vector for problem d could be represented as $w^c = w^d + w^s$, where w^d are task specific parameters and w^s are shared across all tasks. In this framework, all tasks are close to some underlying mean w^s and each one deviates from this mean by w^d. Their SVM style multi-task objective minimizes the loss of w^c and the norm of w^d and w^s, with a tradeoff parameter allowing for domain deviance from the mean. The simple domain adaptation algorithm of feature splitting used by Daumé (2007) is a special case of this model where the norms are equally weighted. An analogous CW objective is:

$$\begin{aligned}\min \frac{1}{\lambda_1}\mathbb{D}_{\text{KL}}\left(\mathcal{N}\left(\boldsymbol{\mu}^d, \Sigma^d\right) \| \mathcal{N}\left(\boldsymbol{\mu}_i^d, \Sigma_i^d\right)\right) \\ + \frac{1}{\lambda_2}\mathbb{D}_{\text{KL}}\left(\mathcal{N}\left(\boldsymbol{\mu}^s, \Sigma^s\right) \| \mathcal{N}\left(\boldsymbol{\mu}_i^s, \Sigma_i^s\right)\right) \\ \text{s.t. } \Pr_{\boldsymbol{w}\sim\mathcal{N}(\boldsymbol{\mu}^c, \Sigma^c)}\left[y_i\left(\boldsymbol{w}\cdot\boldsymbol{x}_i\right) \geq 0\right] \geq \eta\,. \end{aligned} \quad (3)$$

$\left(\boldsymbol{\mu}^d, \Sigma^d\right)$ are the parameters for domain d, $(\boldsymbol{\mu}^s, \Sigma^s)$ for the shared classifier and $(\boldsymbol{\mu}^c, \Sigma^c)$ for the combination of the domain and shared classifiers. The parameters are combined via (2) with only two elements summed - one for the shared parameters s and the other for the domain parameters d . This captures the intuition of Evgeniou and Pontil: updates enforce the learning condition on the combined parameters and minimize parameter change. For convenience, we rewrite $\lambda_2 = 2 - 2\lambda_1$, where $\lambda_1 \in [0, 1]$. If classifiers are combined using the sum of the individual weight vectors and $\lambda_1 = 0.5$, this is identical to feature splitting (Daumé) for CW classifiers.

The domain specific and shared classifiers can be updated using the closed form solution to (3) as:

$$\begin{aligned}
\boldsymbol{\mu}^s &= \boldsymbol{\mu}_i^s + \lambda_2 \alpha y_i \Sigma^c \boldsymbol{x}_i \\
(\Sigma^s)^{-1} &= (\Sigma_i^s)^{-1} + 2\lambda_2 \alpha \phi \boldsymbol{x}_i \boldsymbol{x}_i^T \\
\boldsymbol{\mu}^d &= \boldsymbol{\mu}_i^d + \lambda_1 \alpha y_i \Sigma_i^c \boldsymbol{x}_i \\
(\Sigma^d)^{-1} &= (\Sigma_i^d)^{-1} + 2\lambda_1 \alpha \phi \boldsymbol{x}_i \boldsymbol{x}_i^T
\end{aligned} \tag{4}$$

We call this objective Multi-Domain Regularization (MDR). As before, the combined parameters are produced by one of the combination methods. On each round, the algorithm receives instance $\boldsymbol{x}_i$ and domain d_i for which it creates a combined classifier $(\boldsymbol{\mu}^c, \Sigma^c)$ using the shared $(\boldsymbol{\mu}^s, \Sigma^s)$ and domain specific parameters $\left(\boldsymbol{\mu}^d, \Sigma^d\right)$. A prediction is issued using the standard linear classifier prediction rule $\text{sign}(\boldsymbol{\mu}^c \cdot \boldsymbol{x})$ and updates follow (4). The effect is that features similar across domains quickly converge in the shared classifier, sharing information across domains. The combined classifier reflects shared and domain specific parameter confidences: weights with low variance (i.e. greater confidence) will contribute more.

We evaluate MDR on a single pass over a stream of instances from multiple domains, simulating a real world setting. Parameters λ_1 and ϕ are iteratively optimized on a single randomized run for each dataset. All experiments use 10-fold CV. In addition to evaluating the four combination methods with MDR, we evaluate the performance of a single classifier trained on all domains (*Single*), a separate classifier trained on each domain (*Separate*), *Feature Splitting* (Daumé) and feature splitting with optimized λ_1 (*MDR*). Table 3 shows results on test data and table 2 shows online training error.

In this setting, L2 combinations prove best on 5 of 6 datasets, with the variance weighted combination doing the best. MDR (optimizing λ_1) slightly improves over feature splitting, and the combination methods improve in every case. Our best result is statistically significant compared to *Feature Splitting* using McNemar's test ($p = .001$) for *Task B*, *Books*, *DVD*, *Books+DVD*. While a single or separate classifiers have a different effect on each dataset, MDR gives the best performance overall.

8 Learning in Many Domains

So far we have considered settings with a small number of similar domains. While this is typical of multi-task problems, real world settings present many domains which do not all share the same behaviors. Online algorithms scale to numerous examples and we desire the same behavior for numerous domains. Consider a spam filter used by a large email provider, which filters billions of emails for millions of users. Suppose that spammers control many accounts and maliciously label spam as legitimate. Alternatively, subsets of users may share preferences. Since behaviors are not consistent across domains, shared parameters cannot be learned. We seek algorithms robust to this behavior.

Since subsets of users share behaviors, these can be learned using our MDR framework. For example, discovering spammer and legitimate mail accounts would enable intra-group learning. The challenge is the online discovery of these subsets while learning model parameters. We augment the MDR framework to additionally learn this mapping.

We begin by generalizing MDR to include k shared classifiers instead of a single set of shared parameters. Each set of shared parameters represents a different subset of domains. If the corresponding shared parameters are known for a domain, we could use the same objective (3) and update (4) as before. If there are many fewer shared parameters than domains ($k \ll D$), we can benefit from multi-domain learning. Next, we augment the learning algorithm to learn a mapping between the domains and shared classifiers. Intuitively, a domain should be mapped to shared parameters that correctly classify that domain. A common technique for learning such experts in the Weighted Majority algorithm (Littlestone and Warmuth, 1994), which weighs a mixture of experts (classifiers). However, since we require a hard assignment — pick a single shared parameter set s — rather than a mixture, the algorithm reduces to picking the classifier s with the fewest mistakes in predicting domain d. This requires tracking the number of mistakes made by each shared classifier on each domain once a label is revealed. For learning, the shared classifier with the fewest mistakes for a domain is selected for an MDR update. Classifier ties are broken randomly. While we experi-

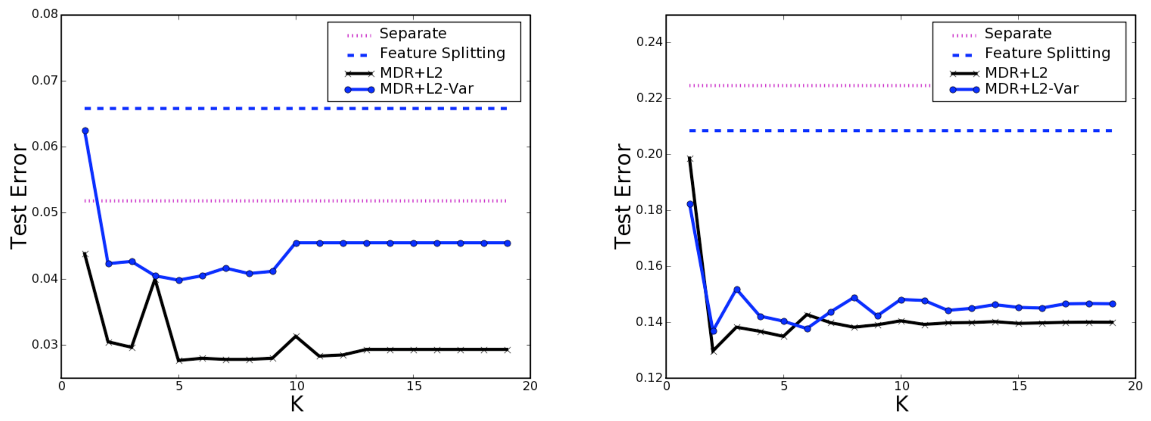


Figure 1: Learning across many domains - spam (left) and sentiment (right) - with MDR using k shared classifiers.

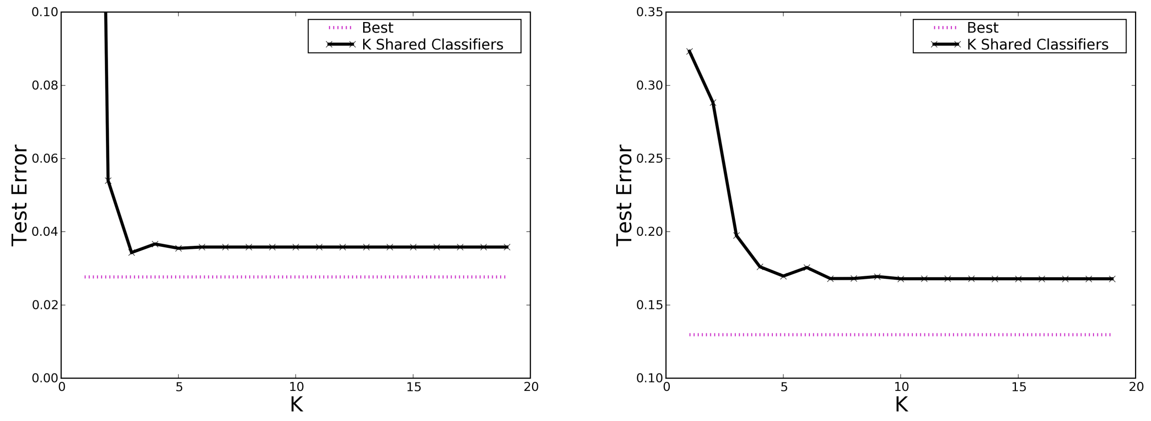


Figure 2: Learning across many domains - spam (left) and sentiment (right) - with no domain specific parameters.

mented with more complex techniques, this simple method worked well in practice. When a new domain is added to the system, it takes fewer examples to learn which shared classifier to use instead of learning a new model from scratch.

While this approach adds another free parameter (k) that can be set using development data, we observe that k can instead be fixed to a large constant. Since only a single shared classifier is updated each round, the algorithm will favor selecting a previously used classifier as opposed to a new one, using as many classifiers as needed but not scaling up to k. This may not be optimal, but it is a simple.

To evaluate a larger number of domains, we created many varying domains using spam and sentiment data. For spam, 6 email users were created by splitting the 3 task A users into 2 users, and flipping the label of one of these users (a malicious user), yielding 400 train and 100 test emails per user. For sentiment, the book domain was split into 3 groups with binary boundaries at a rating of 2, 3 or 4. Each of these groups was split into 8 groups of which half had their labels flipped, creating 24 domains. The same procedure was repeated for DVD reviews but for a decision boundary of 3, 6 groups were created, and for a boundary of 2 and 4, 3 groups were created with 1 and 2 domains flipped respectively, resulting in 12 DVD domains and 36 total domains with various decision boundaries, features, and inverted decision functions. Each domain used 300 train and 100 test instances. 10-fold cross validation with one training iteration was used to train models on these two datasets. Parameters were optimized as before. Experiments were repeated for various settings of k. Since L2 performed well before, we evaluated MDR+L2 and MDR+L2-Var.

The results are shown in figure 1. For both spam and sentiment adding additional shared parameters beyond the single shared classifier significantly reduces error, with further reductions as k increases. This yields a 45% error reduction for spam and a 38% reduction for sentiment over the best baseline. While each task has an optimal k (about 5 for spam, 2 for sentiment), larger values still achieve low error, indicating the flexibility of using large k values.

While adding parameters clearly helps for many domains, it may be impractical to keep domain-specific classifiers for thousands or millions of domains. In this case, we could eliminate the domain-specific classifiers and rely on the k shared classifiers only, learning the domain to classifier mapping. We compare this approach using the best result from MDR above, again varying k. Figure 2 shows that losing domain-specific parameters hurts performance, but is still an improvement over baseline methods. Additionally, we can expect better performance as the number of similar domains increases. This may be an attractive alternative to keeping a very large number of parameters.

9 Related Work

Multi-domain learning intersects two areas of research: domain adaptation and multi-task learning. In domain adaptation, a classifier trained for a source domain is transfered to a target domain using either unlabeled or a small amount of labeled target data. Blitzer et al. (2007) used structural correspondence learning to train a classifier on source data with new features induced from target unlabeled data. In a complimentary approach, Jiang and Zhai (2007) weighed training instances based on their similarity to unlabeled target domain data. Several approaches utilize source data for training on a limited number of target labels, including feature splitting (Daumé, 2007) and adding the source classifier's prediction as a feature (Chelba and Acero, 2004). Others have considered transfer learning, in which an existing domain is used to improve learning in a new domain, such as constructing priors (Raina et al., 2006; Marx et al., 2008) and learning parameter functions for text classification from related data (Do and Ng, 2006). These methods largely require batch learning, unlabeled target data, or available source data at adaptation. In contrast, our algorithms operate purely online and can be applied when no target data is available.

Multi-task algorithms, also known as inductive transfer, learn a set of related problems simultaneously (Caruana, 1997). The most relevant approach is that of Regularized Multi-Task Learning (Evgeniou and Pontil, 2004), which we use to motivate our online algorithm. Dekel et al. (2006) gave a similar online approach but did not use shared parameters and assumed multiple instances for each round. We generalize this work to both include an arbitrary classifier combination and many shared classifiers. Some multi-task work has also considered the grouping of tasks similar to our learning of domain subgroups (Thrun and O'Sullivan, 1998; Bakker and Heskes, 2003).

There are many techniques for combining the output of multiple classifiers for ensemble learning or mixture of experts. Kittler et al. (Mar 1998) provide a theoretical framework for combining classifiers. Some empirical work has considered adding versus multiplying classifier output (Tax et al., 2000), using local accuracy estimates for combination (Woods et al., 1997), and applications to NLP tasks (Florian et al., 2003). However, these papers consider combining classifier output for prediction. In contrast, we consider parameter combination for both prediction and learning.

10 Conclusion

We have explored several multi-domain learning settings using CW classifiers and a combination method. Our approach creates a better classifier for a new target domain than selecting a random source classifier a prior, reduces learning error on multiple domains compared to baseline approaches, can handle many disparate domains by using many shared classifiers, and scales to a very large number of domains with a small performance reduction. These scenarios are realistic for NLP systems in the wild. This work also raises some questions about learning on large numbers of disparate domains: can a hierarchical online clustering yield a better representation than just selecting between k shared parameters? Additionally, how can prior knowledge about domain similarity be included into the combination methods? We plan to explore these questions in future work.

Acknowledgements This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-07-D-0185.

References

B. Bakker and T. Heskes. 2003. Task clustering and gating for bayesian multi–task learning. *Journal of Machine Learning Research*, 4:83–99.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Association for Computational Linguistics (ACL)*.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28:41–75.

Ciprian Chelba and Alex Acero. 2004. Adaptation of max- imum entropy classifier: Little data can help a lot. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Hal Daumé. 2007. Frustratingly easy domain adaptation. In *Association for Computational Linguistics (ACL)*.

Ofer Dekel, Philip M. Long, and Yoram Singer. 2006. Online multitask learning. In *Conference on Learning Theory (COLT)*.

Chuong B. Do and Andrew Ng. 2006. Transfer learning for text classification. In *Advances in Neural Information Processing Systems (NIPS)*.

Mark Dredze, Koby Crammer, and Fernando Pereira. 2008. Confidence-weighted linear classification. In *International Conference on Machine Learning (ICML)*.

Theodoros Evgeniou and Massimiliano Pontil. 2004. Regularized multi-task learning. In *Conference on Knowledge Discovery and Data Mining (KDD)*.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Conference on Computational Natural Language Learning (CONLL)*.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Association for Computational Linguistics (ACL)*.

J. Kittler, M. Hatef, R.P.W. Duin, and J. Matas. Mar 1998. On combining classifiers. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(3):226–239.

N. Littlestone and M. K. Warmuth. 1994. The weighted majority algorithm. *Information and Computation*, 108:212–261.

Zvika Marx, Michael T. Rosenstein, Thomas G. Dietterich, and Leslie Pack Kaelbling. 2008. Two algorithms for transfer learning. In *Inductive Transfer: 10 years later*.

Rajat Raina, Andrew Ng, and Daphne Koller. 2006. Constructing informative priors using transfer learning. In *International Conference on Machine Learning (ICML)*.

David M. J. Tax, Martijn van Breukelen, Robert P. W. Duina, and Josef Kittler. 2000. Combining multiple classifiers by averaging or by multiplying? *Pattern Recognition*, 33(9):1475–1485, September.

S. Thrun and J. O'Sullivan. 1998. Clustering learning tasks and the selective cross–task transfer of knowledge. In S. Thrun and L.Y. Pratt, editors, *Learning To Learn*. Kluwer Academic Publishers.

Kevin Woods, W. Philip Kegelmeyer Jr., and Kevin Bowyer. 1997. Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):405–410.

Jointly Combining Implicit Constraints Improves Temporal Ordering

Nathanael Chambers and Dan Jurafsky
Department of Computer Science
Stanford University
Stanford, CA 94305
{natec,jurafsky}@stanford.edu

Abstract

Previous work on ordering events in text has typically focused on local pairwise decisions, ignoring globally inconsistent labels. However, temporal ordering is the type of domain in which global constraints should be relatively easy to represent and reason over. This paper presents a framework that informs local decisions with two types of implicit global constraints: transitivity (*A before B* and *B before C* implies *A before C*) and time expression normalization (e.g. *last month* is before *yesterday*). We show how these constraints can be used to create a more densely-connected network of events, and how global consistency can be enforced by incorporating these constraints into an integer linear programming framework. We present results on two event ordering tasks, showing a 3.6% absolute increase in the accuracy of *before/after* classification over a pairwise model.

1 Introduction

Being able to temporally order events is a necessary component for complete document understanding. Interest in machine learning approaches for this task has recently been encouraged through the creation of the Timebank Corpus (Pustejovsky et al., 2003). However, most work on event-event ordering has focused on improving classifiers for pairwise decisions, ignoring obvious contradictions in the global space of events when misclassifications occur. A global framework to repair these event ordering mistakes has not yet been explored.

This paper addresses three main factors involved in a global framework: the global optimization algorithm, the constraints that are relevant to the task, and the level of connectedness across pairwise decisions. We employ Integer Linear Programming to address the first factor, drawing from related work in paragraph ordering (Bramsen et al., 2006). After finding minimal gain with the initial model, we explore reasons for and solutions to the remaining two factors through temporal reasoning and transitivity rule expansion.

We analyze the connectivity of the Timebank Corpus and show how textual events can be indirectly connected through a time normalization algorithm that automatically creates new relations between time expressions. We show how this increased connectivity is essential for a global model to improve performance.

We present three progressive evaluations of our global model on the Timebank Corpus, showing a 3.6% gain in accuracy over its original set of relations, and an 81% increase in training data size from previous work. In addition, we present the first results on Timebank that include an *unknown* relation, establishing a benchmark for performance on the full task of document ordering.

2 Previous Work

Recent work on classifying temporal relations within the Timebank Corpus built 6-way relation classifiers over 6 of the corpus' 13 relations (Mani et al., 2006; Mani et al., 2007; Chambers et al., 2007). A wide range of features are used, ranging from surface indicators to semantic classes. Classifiers make local pairwise decisions and do not consider global implications between the relations.

The TempEval-07 (Verhagen et al., 2007) contest recently used two relations, *before* and *after*, in a semi-complete textual classification task with a new third relation to distinguish relations that can be labeled with high confidence from those that are uncertain, called *vague*. The task was a simplified classification task from Timebank in that only one verb, the main verb, of each sentence was used. Thus, the task can be viewed as ordering the main events in pairwise sentences rather than the entire document.

This paper uses the core relations of TempEval (*before*,*after*,*vague*) and applies them to a full document ordering task that includes every labeled event in Timebank. In addition, we extend the previous work by including a temporal reasoning component and embedding it within a global constraint model.

3 The Timebank Corpus

The Timebank Corpus (Pustejovsky et al., 2003) is a corpus of 186 newswire articles that are tagged for events, time expressions, and relations between the events and times. The individual events are further tagged for temporal information such as tense, modality and grammatical aspect. Time expressions use the TimeML (Ingria and Pustejovsky, 2002) markup language. There are 6 main relations and their inverses in Timebank: *before*, *ibefore*, *includes*, *begins*, *ends* and *simultaneous*.

This paper describes work that classifies the relations between events, making use of relations between events and times, and between the times themselves to help inform the decisions.

4 The Global Model

Our initial model has two components: (1) a pairwise classifier between events, and (2) a global constraint satisfaction layer that maximizes the confidence scores from the classifier. The first is based on previous work (Mani et al., 2006; Chambers et al., 2007) and the second is a novel contribution to event-event classification.

4.1 Pairwise Classification

Classifying the relation between two events is the basis of our model. A soft classification with confidence scores is important for the global maximization step that is described in the next section. As in Chambers et al. (2007), we build support vector machine (SVM) classifiers and use the probabilities from pairwise SVM decisions as our confidence scores. These scores are then used to choose an optimal global ordering.

Following our previous work, we use the set of features summarized in figure 1. They vary from POS tags and lexical features surrounding the event, to syntactic dominance, to whether or not the events share the same tense, grammatical aspect, or aspectual class. These features are the highest performing set on the basic 6-way classification of Timebank.

Feature	Description
Word*	The text of the event
Lemma*	The lemmatized head word
Synset*	The WordNet synset of head word
POS*	4 POS tags, 3 before, and 1 event
POS bigram*	The POS bigram of the event and its preceding tag
Prep*	Preposition lexeme, if in a prepositional phrase
Tense*	The event's tense
Aspect*	The event's grammatical aspect
Modal*	The modality of the event
Polarity*	Positive or negative
Class*	The aspecual class of the event
Tense Pair	The two concatenated tenses
Aspect Pair	The two concatenated aspects
Class Pair	The two concatenated classes
POS Pair	The two concatenated POS tags
Tense Match	true if the events have the same tense
Aspect Match	true if the events have the same aspect
Class Match	true if the events have the same class
Dominates	true if the first event syntactically dominates the second
Text Order	true if the first event occurs first in the document
Entity Match	true if they share an entity as an argument
Same Sent	true if both events are in the same sentence

Figure 1: The features to learn temporal relations between two events. Asterisks (*) indicate features that are duplicated, one for each of the two events.

We use Timebank's hand tagged attributes in the feature values for the purposes of this comparative

	before	after	unknown
A r1 B	.5	.3	.2
B r2 C	.4	.3	.3
A r3 C	.4	.5	.1
total	**1.3**	1.1	.6
A r1 B	.5	.3	.2
B r2 C	.4	.3	.3
A r3 C	.2	.7	.1
total	1.1	**1.3**	.6

Figure 2: Two sets of confidence scores. The first set chooses *before* for all three labels, and the second chooses *after*. Other lower-scoring valid relation sets also exist, such as *before*, *unknown*, and *before*.

study of global constraints, described next.

4.2 Global Constraints

Pairwise classifiers can make contradictory classifications due to their inability to consider other decisions. For instance, the following three decisions are in conflict:

```
A before B
B before C
A after C
```

Transitivity is not taken into account. In fact, there are several ways to resolve the conflict in this example. Given confidence scores (or probabilities) for each possible relation between the three pairs, we can compute an optimal label assignment. Different scores can lead to different conflict resolutions. Figure 2 shows two resolutions given different sets of scores. The first chooses *before* for all three relations, while the second chooses *after*.

Bramsen et al. (2006) presented a variety of approaches to using transitivity constraints to help inform pairwise decisions. They found that Integer Linear Programming (ILP) performed the best on a paragraph ordering task, consistent with its property of being able to find the optimal solution for a set of constraints. Other approaches are variations on a *greedy* strategy of adding pairs of events one at a time, ordered by their confidence. These can lead to suboptimal configurations, although they are guaranteed to find a solution. Mani et al. (2007) subsequently proposed one of these greedy strategies as well, but published results are not available. We also implemented a greedy best-first strategy, but found ILP outperformed it.

Our Integer Linear Programming framework uses the following objective function:

$$max \sum_i \sum_j p_{ij} x_{ij} \tag{1}$$

with added constraints:

$$\forall i \forall j \;\; x_{ij} \in \{0, 1\} \tag{2}$$

$$\forall i \;\; x_{i1} + x_{i2} + ... + x_{im} = 1 \tag{3}$$

where x_{ij} represents the ith pair of events classified as the jth relation of m relations. Thus, each pair of events generates m variables. Given n pairs of events, there are $n * m$ variables. p_{ij} is the probability of classifying pair i with relation j. Equation 2 (the first constraint) simply says that each variable must be 0 or 1. Equation 3 contains m variables for a single pair of events i representing its m possible relations. It states that one relation must be set to 1 and the rest to 0. In other words, a pair of events cannot have two relations at the same time. Finally, a transitivity constraint is added for all connected pairs i, j, k, for each transitivity condition that infers relation c given a and b:

$$x_{ia} + x_{jb} - x_{kc} <= 1 \tag{4}$$

We generated the set of constraints for each document and used lpsolve[1] to solve the ILP constraint problem.

The transitivity constraints are only effective if the available pairwise decisions constitute a connected graph. If pairs of events are disconnected, then transitivity makes little to no contribution because these constraints are only applicable to connected chains of events.

4.3 Transitive Closure

In order to connect the event graph, we draw on work from (Mani et al., 2006) and apply *transitive closure* to our documents. Transitive closure was first proposed not to address the problem of connected event graphs, but rather to expand the size of training data for relations such as *before*. Timebank is a relatively small corpus with few examples

[1]http://sourceforge.net/projects/lpsolve

Total Event-Event Relations After Closure

	before	*after*
Timebank	592	656
+ closure	3919	3405

Figure 3: The number of event-event relations after transitive closure.

Comparative Results

Training Set	Accuracy
Timebank Pairwise	66.8%
Global Model	66.8%

Figure 4: Using the base Timebank annotated tags for testing, accuracy on before/after tags in the two models.

of each relation. One way of expand the training set is through transitive rules. A few rules are given here:

$A\ simultaneous\ B \wedge A\ before\ C \rightarrow B\ before\ C$
$A\ includes\ B \wedge A\ ibefore\ C \rightarrow B\ before\ C$
$A\ before\ B \wedge A\ ends\ C \rightarrow B\ after\ C$

While the original motivation was to expand the training size of tagged relations, this approach also creates new connections in the graph, replacing previously unlabeled event pairs with their true relations. We adopted this approach and closed the original set of 12 relations to help connect the global constraint model.

4.4 Initial Experiment

The first evaluation of our global temporal model is on the Timebank Corpus over the labeled relations *before* and *after*. We merged *ibefore* and *iafter* into these two relations as well, ignoring all others. We use this task as a reduced evaluation to study the specific contribution of global constraints. We also chose this strict ordering task because it is well defined from a human understanding perspective. Snow et al. (2008) shows that average internet users can make *before/after* decisions with very high confidence, although the distinction with an *unknown* relation is not as clear. An evaluation including *unknown* (or *vague* as in TempEval) is presented later.

We expanded the corpus (prior to selecting the before/after relations) using transitive closure over all 12 relations as described above. Figure 3 shows the increase in data size. The number of *before* and *after* relations increase by a factor of six.

We trained and tested the system with 10-fold cross validation and micro-averaged accuracies. The folds were randomly generated to separate the 186 files into 10 folds (18 or 19 files per fold). The same 10-way split is used for all the evaluations. We used libsvm[2] to implement our SVM classifiers.

Figure 4 shows the results from our ILP model with transitivity constraints. The first row is the baseline pairwise classification trained and tested on the original Timebank relations. The second row gives performance with ILP. The model shows no improvement. The global ILP constraints did affect local decisions, changing 175 of them (out of 7324), but the changes cancelled out and had no affect on overall accuracy.

4.5 Loosely Connected Graph

Why didn't a global model help? The problem lies in the graph structure of Timebank's annotated relations. The Timebank annotators were not required to annotate relations between any particular pair of events. Instead, they were instructed to annotate what seemed appropriate due to the almost insurmountable task of annotating all pairs of events. A modest-sized document of 30 events, for example, would contain $\binom{30}{2} = 435$ possible pairs. Annotators thus marked relations where they deemed fit, most likely between obvious and critical relations to the understanding of the article. The vast majority of possible relations are untagged, thus leaving a large set of unlabeled (and disconnected) *unknown* relations.

Figure 5 graphically shows all relations that are annotated between events and time expressions in one of the shorter Timebank documents. Nodes represent events and times (event nodes start with the letter 'e', times with 't'), and edges represent temporal relations. Solid lines indicate hand annotations, and dotted lines indicate new rules from transitive closure (only one, from event *e4* to time *t14*). As can be seen, the graph is largely disconnected and a global model contributes little information since transitivity constraints cannot apply.

[2]http://www.csie.ntu.edu.tw/˜cjlin/libsvm

Timebank Annotation of wsj_0551

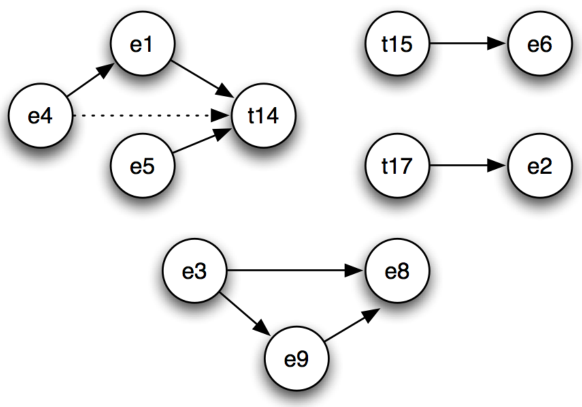


Figure 5: Annotated relations in document wsj_0551.

The large amount of unlabeled relations in the corpus presents several problems. First, building a classifier for these *unknown* relations is easily overwhelmed by the huge training set. Second, many of the untagged pairs have non-*unknown* ordering relations between them, but were missed by the annotators. This point is critical because one cannot filter this noise when training an *unknown* classifier. The noise problem will appear later and will be discussed in our final experiment. Finally, the space of annotated events is very loosely connected and global constraints cannot assist local decisions if the graph is not connected. The results of this first experiment illustrate this latter problem.

Bethard et al. (2007) strengthen the claim that many of Timebank's untagged relations should not be left unlabeled. They performed an independent annotation of 129 of Timebank's 186 documents, tagging all events in verb-clause relationships. They found over 600 valid *before/after* relations that are untagged in Timebank, on average three per document. One must assume that if these nearby verb-clause event pairs were missed by the annotators, the much larger number of pairs that cross sentence boundaries were also missed.

The next model thus attempts to fill in some of the gaps and further connect the event graph by using two types of knowledge. The first is by integrating Bethard's data, and the second is to perform temporal reasoning over the document's time expressions (e.g. *yesterday* or *january 1999*).

5 A Global Model With Time

Our initial model contained two components: (1) a pairwise classifier between events, and (2) a global constraint satisfaction layer. However, due to the sparseness in the event graph, we now introduce a third component addressing connectivity: (3) a temporal reasoning component to inter-connect the global graph and assist in training data expansion.

One important aspect of transitive closure includes the event-time and time-time relations during closure, not just the event-event links. Starting with 5,947 different types of relations, transitive rules increase the dataset to approximately 12,000. However, this increase wasn't enough to be effective in global reasoning. To illustrate the sparsity that still remains, if each document was a fully connected graph of events, Timebank would contain close to 160,000 relations[3], more than a 13-fold increase.

More data is needed to enrich the Timebank event graph. Two types of information can help: (1) more event-event relations, and (2) a separate type of information to indirectly connect the events: event-X-event. We incorporate the new annotations from Bethard et al. (2007) to address (1) and introduce a new temporal reasoning procedure to address (2). The following section describes this novel approach to adding time expression information to further connect the graph.

5.1 Time-Time Information

As described above, we use event-time relations to produce the transitive closure, as well as annotated time-time relations. It is unclear if Mani et al. (2006) used these latter relations in their work.

However, we also add new time-time links that are deduced from the logical time intervals that they describe. Time expressions can be resolved to time intervals with some accuracy through simple rules. New time-time relations can then be added to our space of events through time stamp comparisons. Take this newswire example:

The Financial Times 100-share index shed 47.3 points to close at 2082.1, down 4.5% from the ***previous Friday****, and 6.8% from* ***Oct. 13****, when Wall Street's plunge helped spark the* ***current*** *weakness in London.*

[3]Sum over the # of events n_d in each document d, $\binom{n_d}{2}$

The first two expressions (‘*previous Friday*’ and ‘*Oct. 13*’) are in a clear *before* relationship that Timebank annotators captured. The ‘current’ expression, is correctly tagged with the *PRESENT_REF* attribute to refer to the document’s timestamp. Both ‘*previous Friday*’ and ‘*Oct. 13*’ should thus be tagged as being *before* this expression. However, the annotators did not tag either of these two *before* relations, and so our timestamp resolution procedure fills in these gaps. This is a common example of two expressions that were not tagged by the annotators, yet are in a clear temporal relationship.

We use Timebank’s gold standard TimeML annotations to extract the dates and times from the time expressions. In addition, those marked as *PRESENT_REF* are resolved to the document timestamp. Time intervals that are strictly before or after each other are thus labeled and added to our space of events. We create new *before* relations based on the following procedure:

```
if event1.year < event2.year
  return true
if event1.year == event2.year
  if event1.month < event2.month
    return true
  if event1.month == event2.month
    if event1.day < event2.day
      return true
  end
end
return false
```

All other time-time orderings not including the *before* relation are ignored (i.e. *includes* is not created, although could be with minor changes).

This new time-time knowledge is used in two separate stages of our model. The first is just prior to transitive closure, enabling a larger expansion of our tagged relations set and reduce the noise in the *unknown* set. The second is in the constraint satisfaction stage where we add our automatically computed time-time relations (with the gold event-time relations) to the global graph to help correct local event-event mistakes.

Total Event-Event Relations After Closure

	before	*after*
Timebank	3919	3405
+ time-time	5604	5118
+ time/bethard	7111	6170

Figure 6: The number of event-event *before* and *after* relations after transitive closure on each dataset.

Comparative Results with Closure

Training Set	Accuracy
Timebank Pairwise	66.8%
Global Model	66.8%
Global + time/bethard	70.4%

Figure 7: Using the base Timebank annotated tags for testing, the increase in accuracy on before/after tags.

5.2 Temporal Reasoning Experiment

Our second evaluation continues the use of the two-way classification task with *before* and *after* to explore the contribution of closure, time normalization, and global constraints.

We augmented the corpus with the labeled relations from Bethard et al. (2007) and added the automatically created time-time relations as described in section 5.1. We then expanded the corpus using transitive closure. Figure 6 shows the progressive data size increase as we incrementally add each to the closure algorithm.

The time-time generation component automatically added 2459 new *before* and *after* time-time relations into the 186 Timebank documents. This is in comparison to only 157 relations that the human annotators tagged, less than 1 per document on average. The second row of figure 6 shows the drastic effect that these time-time relations have on the number of available event-event relations for training and testing. Adding both Bethard’s data and the time-time data increases our training set by 81% over closure without it.

We again performed 10-fold cross validation with micro-averaged accuracies, but each fold tested only on the transitively closed Timebank data (the first row of figure 6). The training set used all available data (the third row of figure 6) including the Bethard data as well as our new time-time links.

Figure 7 shows the results from the new model. The first row is the baseline pairwise classification trained and tested on the original relations only. Our model improves by 3.6% absolute. This improvement is statistically significant ($p < 0.000001$, McNemar's test, 2-tailed).

5.3 Discussion

To further illustrate why our model now improves local decisions, we continue our previous graph example. The actual text for the graph in figure 5 is shown here:

docstamp: 10/30/89 ***(t14)***
*Trustcorp Inc. will become****(e1)*** *Society Bank & Trust when its merger****(e3)*** *is completed****(e4)*** *with Society Corp. of Cleveland, the bank said****(e5)****. Society Corp., which is also a bank, agreed****(e6)*** *in June****(t15)*** *to buy****(e8)*** *Trustcorp for 12.4 million shares of stock with a market value of about $450 million. The transaction****(e9)*** *is expected****(e10)*** *to close****(e2)*** *around year end****(t17)****.*

The automatic time normalizer computes and adds three *new* time-time relations, two connecting t15 and t17 with the document timestamp, and one connecting t15 and t17 together. These are not otherwise tagged in the corpus.

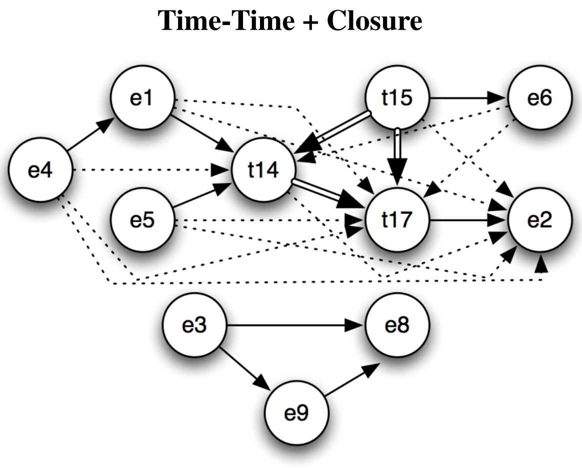


Figure 8: Before and after time-time links with closure.

Figure 8 shows the augmented document. The double-line arrows indicate the three new time-time relations and the dotted edges are the new relations added by our transitive closure procedure. Most critical to this paper, three of the new edges are event-event relations that help to expand our training data. If this document was used in testing (rather than training), these new edges would help inform our transitive rules during classification.

Even with this added information, disconnected segments of the graph are still apparent. However, the 3.6% performance gain encourages us to move to the final full task.

6 Final Experiment with Unknowns

Our final evaluation expands the set of relations to include unlabeled relations and tests on the entire dataset available to us. The following is now a classification task between the three relations: *before*, *after*, and *unknown*.

We duplicated the previous evaluation by adding the labeled relations from Bethard et al. (2007) and our automatically created time-time relations. We then expanded this dataset using transitive closure. Unlike the previous evaluation, we also use this entire dataset for testing, not just for training. Thus, all event-event relations in Bethard as well as Timebank are used to expand the dataset with transitive closure and are used in training and testing. We wanted to fully evaluate document performance on every possible event-event relation that logically follows from the data.

As before, we converted *IBefore* and *IAfter* into *before* and *after* respectively, while all other relations are reduced to *unknown*. This relation set coincides with TempEval-07's core three relations (although they use *vague* instead of *unknown*).

Rather than include all unlabeled pairs in our *unknown* set, we only include the unlabeled pairs that span at most one sentence boundary. In other words, events in adjacent sentences are included in the *unknown* set if they were not tagged by the Timebank annotators. The intuition is that annotators are more likely to label nearby events, and so events in adjacent sentences are more likely to be actual *unknown* relations if they are unlabeled. It is more likely that distant events in the text were overlooked by convenience, not because they truly constituted an *unknown* relationship.

The set of possible sentence-adjacent *unknown* relations is very large (approximately 50000 *unknown* compared to 7000 *before*), and so we randomly select a percentage of these relations for each evalu-

Classification Accuracy

% unk	base	global	global+time
0	72.0%	72.2%	74.0%
1	69.4%	69.5%	71.3%
3	65.5%	65.6%	67.1%
5	63.7%	63.8%	65.3%
7	61.2%	61.6%	62.8%
9	59.3%	59.5%	60.6%
11	58.1%	58.4%	59.4%
13	57.1%	57.1%	58.1%

Figure 9: Overall accuracy when training with different percentages of *unknown* relations included. 13% of *unknowns* is about equal to the number of *befores*.

Base Pairwise Classification

	precision	recall	f1-score
before	61.4	55.4	58.2
after	57.6	53.1	55.3
unk	53.0	62.8	57.5

Global+Time Classification

	precision	recall	f1-score
before	63.7 (+2.3)	57.1 (+2.2)	60.2 (+2.0)
after	60.3 (+2.7)	54.3 (+2.9)	57.1 (+1.8)
unk	52.0 (-1.0)	62.9 (+0.1)	56.9 (-0.6)

Figure 10: Precision and Recall for the base pairwise decisions and the global constraints with integrated time information.

ation. We used the same SVM approach with the features described in section 4.1.

6.1 Results

Results are presented in figure 9. The rows in the table are different training/testing runs on varying sizes of *unknown* training data. There are three columns with accuracy results of increasing complexity. The first, **base**, are results from pairwise classification decisions over Timebank and Bethard with no global model. The second, **global**, are results from the Integer Linear Programming global constraints, using the pairwise confidence scores from the **base** evaluation. Finally, the **global+time** column shows the ILP results when all event-time, time-time, and automatically induced time-time relations are included in the global graph.

The ILP approach does not alone improve performance on the event-event tagging task, but adding the time expression relations greatly increases the global constraint results. This is consistent with the results from out first two experiments. The evaluation with 1% of the *unknown* tags shows an almost 2% improvement in accuracy. The gain becomes smaller as the *unknown* set increases in size (1.0% gain with 13% *unknown*). *Unknown* relations will tend to be chosen as more weight is given to *unknowns*. When there is a constraint conflict in the global model, *unknown* tends to be chosen because it has no transitive implications. All improvements from base to global+time are statistically significant ($p < 0.000001$, McNemar's test, 2-tailed).

The first row of figure 9 corresponds to the results in our second experiment in figure 7, but shows higher accuracy. The reason is due to our different test sets. This final experiment includes Bethard's event-event relations in testing. The improved performance suggests that the clausal event-event relations are easier to classify, agreeing with the higher accuracies originally found by Bethard et al. (2007).

Figure 10 shows the precision, recall, and f-score for the evaluation with 13% *unknowns*. This set was chosen for comparison because it has a similar number of *unknown* labels as *before* labels. We see an increase in precision in both the *before* and *after* decisions by up to 2.7%, an increase in recall up to 2.9%, and an fscore by as much as 2.0%. The *unknown* relation shows mixed results, possibly due to its noisy behavior as discussed throughout this paper.

6.2 Discussion

Our results on the two-way (before/after) task show that adding additional implicit temporal constraints and then performing global reasoning results in significant improvements in temporal ordering of events (3.6% absolute over simple pairwise decisions).

Both *before* and *after* also showed increases in precision and recall in the three-way evaluation. However, *unknown* did not parallel this improvement, nor are the increases as dramatic as in the two-way evaluation. We believe this is consistent with the noise that exists in the Timebank corpus for unlabeled relations. Evidence from Bethard's independent annotations directly point to missing relations, but the dramatic increase in the size of our closure data (81%) from adding a small amount of time-time relations suggests that the problem is widespread. This noise in the *unknown* relation may be dampening the gains that the two way task illustrates.

This work is also related to the task of event-time classification. While not directly addressed in this paper, the global methods described within clearly apply to pairwise models of event-time ordering as well.

Further progress in improving global constraints will require new methods to more accurately identify *unknown* events, as well as new approaches to create implicit constraints over the ordering. We expect such an improved ordering classifier to be used to improve the performance of tasks such as summarization and question answering about the temporal nature of events.

Acknowledgments

This work is funded in part by DARPA through IBM and by the DTO Phase III Program for AQUAINT. We also thank our anonymous reviewers for many helpful suggestions.

References

Steven Bethard, James H. Martin, and Sara Klingenstein. 2007. Timelines from text: Identification of syntactic temporal relations. In *International Conference on Semantic Computing*.

Philip Bramsen, Pawan Deshpande, Yoong Keok Lee, and Regina Barzilay. 2006. Inducing temporal graphs. In *Proceedings of EMNLP-06*.

Nathanael Chambers, Shan Wang, and Dan Jurafsky. 2007. Classifying temporal relations between events. In *Proceedings of ACL-07*, Prague, Czech Republic.

R Ingria and James Pustejovsky. 2002. TimeML specification 1.0. In *http://www.time2002.org*.

Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. In *Proceedings of ACL-06*, July.

Inderjeet Mani, Ben Wellner, Marc Verhagen, and James Pustejovsky. 2007. Three approaches to learning tlinks in timeml. Technical Report CS-07-268, Brandeis University.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, David Day, Lisa Ferro, Robert Gaizauskas, Marcia Lazo, Andrea Setzer, and Beth Sundheim. 2003. The timebank corpus. *Corpus Linguistics*, pages 647–656.

Rion Snow, Brendan O'Connor, Dan Jurafsky, and Andrew Ng. 2008. Cheap and fast - but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of EMNLP-08*, Waikiki, Hawaii, USA.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. Semeval-2007 task 15: Tempeval temporal relation identification. In *Workshop on Semantic Evaluations*.

Automatic inference of the temporal location of situations in Chinese text

Nianwen Xue
Center for Computational Language and Education Research
University of Colorado at Boulder
Colorado, U.S.A.
Nianwen.Xue@colorado.edu

Abstract

Chinese is a language that does not have morphological tense markers that provide explicit grammaticalization of the temporal location of situations (events or states). However, in many NLP applications such as Machine Translation, Information Extraction and Question Answering, it is desirable to make the temporal location of the situations explicit. We describe a machine learning framework where different sources of information can be combined to predict the temporal location of situations in Chinese text. Our experiments show that this approach significantly outperforms the most frequent tense baseline. More importantly, the high training accuracy shows promise that this challenging problem is solvable to a level where it can be used in practical NLP applications with more training data, better modeling techniques and more informative and generalizable features.

1 Introduction

In a language like English, tense is an explicit (and maybe imperfect) grammaticalization of the temporal location of situations, and such temporal location is either directly or indirectly defined in relation to the moment of speech. Chinese does not have grammaticalized tense in the sense that Chinese verbs are not morphologically marked for tense. This is not to say, however, that Chinese speakers do not attempt to convey the temporal location of situations when they speak or write, or that they cannot interpret the temporal location when they read Chinese text, or even that they have a different way of representing the temporal location of situations. In fact, there is evidence that the temporal location is represented in Chinese in exactly the same way as it is represented in English and most world languages: in relation to the moment of speech. One piece of evidence to support this claim is that Chinese temporal expressions like 今天 ("today"), 明天 ("tomorrow") and 昨天 ("yesterday") all assume a temporal deixis that is the moment of speech in relation to which all temporal locations are defined. Such temporal expressions, where they are present, give us a clear indication of the temporal location of the situations they are associated with. However, not all Chinese sentences have such temporal expressions associated with them. In fact, they occur only infrequently in Chinese text. It is thus theoretically interesting to ask, in the absence of grammatical tense and explicit temporal expressions, how do readers of a particular piece of text interpret the temporal location of situations?

There are a few linguistic devices in Chinese that provide obvious clues to the temporal location of situations, and one such linguistic device is aspect markers. Although Chinese does not have grammatical tense, it does have grammaticalized aspect in the form of aspect markers. These aspect markers often give some indication of the temporal location of an event. For example, Chinese has the perfective aspect marker 了 and 过, and they are often associated with the past. Progressive aspect marker 着, on the other hand, is often associated with the present. In addition to aspect, certain adverbs also provide clues to the temporal location of the situations they are associated with. For example, 已 or 已经 (”already”), often indicates that the situation they are associated with has already occurred and is thus in the past. 在, another adverbial modifier, often indicates that the situation it modifies is in the present. However, such linguistic associations are imperfect, and they can only be viewed as tendencies rather than rules that one can use to deterministically infer the temporal location of a situation. For example, while 已 indeed indicates that the situation described in (1) is in the past, when it modifies a stative verb as it does in (1b), the situation is still in the present.

(1) a. 他 [已] 做完 该 项目 。
he already finish this project .

”He already finished the project.”

b. 中国 [已] 拥有 产生 世界级
China already has produce world-class
软件 的 基础 。
software DE foundation .

”China already has the foundation to produce world-class software.”

More importantly, only a small proportion of verb instances in any given text have such explicit temporal indicators and therefore they cannot be the whole story in the temporal interpretation of Chinese text. It is thus theoretically interesting to go beyond the obvious and investigate what additional information is relevant in determining the temporal location of a situation in Chinese.

Being able to infer the temporal location of a situation has many practical applications as well. For example, this information would be highly valuable to Machine Translation. To translate a language like Chinese into a language like English in which tense is grammatically marked with inflectional morphemes, an MT system will have to infer the necessary temporal information to determine the correct tense for verbs. Statistical MT systems, the currently dominant research paradigm, typically do not address this issue directly. As a result, when evaluated for tense, current MT systems often perform miserably. For example, when a simple sentence like “他/he 明天/tomorrow 返回/return 上海/Shanghai” is given to Google’s state-of-the-art Machine Translation system[1], it produces the output “He returned to Shanghai tomorrow”, instead of the correct “he will return to Shanghai tomorrow”. The past tense on the verb “returned” contradicts the temporal expression “tomorrow”. Determining the temporal location is also important for an Information Extraction task that extracts events so that the extracted events are put in a temporal context. Similarly, for Question Answering tasks, it is also important to know whether a situation has already happened or it is going to happen, for example.

In this paper, we are interested in investigating the kind of information that is relevant in inferring the temporal location of situations in Chinese text. We approach this problem by manually annotating each verb in a Chinese document with a “tense” tag that indicates the temporal location of the verb[2]. We then formulate the tense determination problem as a classification task where standard machine learning techniques can be applied. Figuring out what linguistic information contributes to the determination of the temporal location of a situation becomes a feature engineering problem of selecting features that help with the automatic classification. In Section 2, we present a linguistic annotation framework that annotates the temporal location of situations in Chinese text. In Section 3 we describe our setup for an automatic tense classification experiment and present our experimental results. In Section 4 we focus in on the features we have used in our experiment and attempt to provide a quantitative as well as intuitive explanation of the contribution of the individual features and speculate on what additional features could be useful. In Section 5 we discuss related work and Section 6 concludes the paper and discusses future work.

2 Annotation framework

It is impossible to define the temporal location without a reference point, a temporal deixis. As we have shown in Section 1, there is convincing evidence from the temporal adverbials like 昨天(“yesterday”), 今天(“today”) and 明天

[1]http://www.google.com/language_tools

[2]For simplicity, we use the term “tense” exchangeably with the temporal location of an event or situation, even though tense usually means grammatical tense while temporal location is a more abstract semantic notion.

(“tomorrow”) that Chinese, like most if not all languages of the world, use the moment of speech as this reference point. In written text, which is the primary source of data that we are dealing with, the temporal deixis is the document creation time. All situations are temporally related to this document creation time except in direct quotations, where the temporal location is relative to the moment of speech of the speaker who is quoted.

In addition to the moment of speech or document creation time in the case of written text, Reference Time and Situation Time are generally accepted as important to determining the temporal location since Reichenbach (1947) first proposed them. Situation Time is the time that a situation actually occurs while Reference time is the temporal perspective from which the speaker invites his audience to consider the situation. Reference Time does not necessarily overlap with Situation Time, as in the case of present perfective tense, where the situation happened in the past but the reader is invited to look at it from the present moment and focus on the state of completion of the situation. Reference Time is in our judgment too subtle to be annotated consistently and thus in our annotation scheme we only consider the relation between Situation Time and the document creation time when defining the temporal location of situations. Another key decision we made when formulating our annotation scheme is to define an abstract “tense” that do not necessarily model the actual tense system in any particular language that has grammatical tense. In a given language, the grammatical tense reflected in the morphological system may not have a one-to-one mapping between the grammatical tense and the temporal location of a situation. For example, in an English sentence like “He will call me after he gets here”, while his “getting here” happens at a time in the future, it is assigned the present tense because it is in a clause introduced by “after”. It makes more sense to ask the annotator, who is necessarily a native speaker of Chinese, to make a judgment of the temporal location of the situation defined in terms of the relation between the Situation Time and the moment of speech rather than by such language-specific idiosyncracies of another language.

Temporal locations that can be defined in terms of the relation between Situation Time and the moment of speech are considered to be *absolute tense*. In some cases, the temporal location of a situation cannot be directly defined in relation to the moment of speech. For example in (2), the temporal location of 有意 (“intend”) cannot be determined independently of that of 透露(“reveal”). The temporal location of 有意 is simultaneous with 透露. If the temporal location of 透露 is in the past, then the temporal location of 有意 is also in the past. If the temporal location of 透露 is in the future, then the temporal location of 有意 is also in the future. In this specific case, the situation denoted by the matrix verb 透露 is in the past. Therefore the situation denoted by 有意 is also located in the past.

(2) 他 还 透露 俄罗斯 有意 在 今后 十年
he also reveal Russia intend in next ten years
内 , 向 伊朗 提供 武器 .
within , to Iran provide weapons .

“He also revealed that Russia intended to provide weapons to Iran within the next ten years.”

Therefore in our Chinese “tense” annotation task, we annotate both *absolute* and *relative* tenses. We define three absolute tenses based on whether the situation time is anterior to (in the past), simultaneous with (in the present), or posterior to (in the future) document creation time. In addition to the absolute tenses, we also define one relative tense, future-in-past, which happens when a future situation is embedded in a past context. We do not assign a tense tag to modal verbs or verb particles. The set of tense tags are described in more detail below:

2.1 Present tense

A situation is assigned the present tense if it is true at an interval of time that includes the present moment. The present tense is compatible with states and activities. When non-stative situations are temporally located in the present, they either have an imperfective aspect or have a habitual or frequentive reading which makes them look like states, e.g.,

(3) 他 常常 参加 户外 活动 。
he often attend outdoors activities .

“He often attends outdoors activities.”

2.2 Past tense

Situations that happen before the moment of speech (or the document creation time) are temporally located in the past as in (4):

(4) 中方 人员 及 侨胞
Chinese personnel and Chinese nationals
安全 撤离 乍得 。
safely withdraw from Chad .

"Chinese personnel and Chinese nationals safely withdrew from Chad."

2.3 Future tense

Situations that happen posterior to the moment of speech are temporally located in the future. Future situations are not simply the opposite of past situations. While past situations have already happened by definition, future situations by nature are characterized by uncertainty. That is, future situations may or may not happen. Therefore, future situations are often linked to possibilities, not just to situations that will definitely happen. A example of future tense is given in (5):

(5) 大会 明年 在新加坡 举行 。
conference next year in Singapore hold .

"The conference will be held in Singapore next year."

2.4 Future-in-past

The temporal interpretation of one situation is often bound by the temporal location of another situation. One common scenario in which this kind of dependency occurs is when the target situation, the situation we are interested in at the moment, is embedded in a reference situation as its complement. Just as the absolute "tense" represents a temporal relation between the situation time and the moment of speech or document creation time, the relative "tense" represents a relation between the temporal location of a situation and its reference situation. Although theoretically the target situation can be anterior to, simultaneous with, or posterior to the reference situation, we only have a special tense label when the target situation is posterior to the reference situation and the reference situation is located in the past. In this case the label for the target situation is future-in-past as illusrated in (6):

(6) 公司 员工 透露 《星际2》测试
company personnel reveal " Star 2 " trial
版 即将面世 。
version soon face the world .

"The company personnel revealed that 'Star 2' trial version would soon face the world."

2.5 No tense label

Modals and verb particles do not receive a tense label:

(7) 科索沃 独立 可能 引发 骚乱,
Kosovo independence may cause riot .
联合国人员 已 准备 撤离 。
UN personnel already prepare withdraw .

"Kosovo independence may cause riot. UN personnel have already prepared to leave."

The "situations" that we are interested in are expressed as clauses centered around a verb, and for the sake of convenience we mark the "tense" on the verb itself instead of the entire clause. However, when inferring the temporal location of a situation, we have to take into consideration the entire clause, because the arguments and modifiers of a verb are just as important as the verb itself when determining the temporal location of the situation. The annotation is performed on data selected from the Chinese Treebank (Xue et al., 2005), and more detailed descriptions and justifications for the annotation scheme is described in (Xue et al., 2008). Data selection is important for tense annotation because, unlike POS-tagging and syntactic annotation, which applies equally well to different genres of text, temporal annotation in more relevant in some genres than others. The data selection task is made easier by the fact that the Chinese Treebank is already annotated with POS tags and Penn Treebank-style syntactic structures. Therefore we were able to just select articles based on how many constituents in the article are annotated with the temporal function tag -TMP. We have annotated 42 articles in total, and all verbs in an article are assigned one of the five tags described above: present, past, future, future-in-past, and none.

3 Experimental results

The tense determination task is then a simple five-way classification task. Theoretically any standard machine learning algorithm can be applied to the task. For our purposes we used the Maximum Entropy algorithm implemented as part of the Mallet machine learning package (McCallum, 2002) for its competitive training time and performance tradeoff. There might be algorithms that could achieve higher classification accuracy, but our goal in this paper is not to pursue the absolute high performance. Rather, our purpose is to investigate what information when used as features is relevant to determining the temporal location of a situation in Chinese, so that these features can be used to design high performance practical systems in the future.

The annotation of 42 articles yielded 5709 verb instances, each of which is annotated with one of the five tense tags. For our automatic classification experiments, we randomly divided the data into a training set and a test set based on a 3-to-1 ratio, so that the training data has 4,250 instances while the test set has 1459 instances. As expected, the past tense is the most frequent tense in both the training and test data, although they vary quite a bit in the proportions of verbs that are labeled with the past tense. In the training data, 2145, or 50.5% of the verb instances are labeled with the past tense while in the test data, 911 or 62.4% of the verb instances are labeled with the past tense. The 62.4% can thus be used as a baseline when evaluating the automatic classification accuracy. This is a very high baseline given that the much smaller proportion of verbs that are assigned the past tense in the training data.

Instead of raw text, the input to the classification algorithm is parsed sentences from the Chinese Treebank that has the syntactic structure information as well as the part-of-speech tags. As we will show in the next section, information extracted from the parse tree as well as the part-of-speech tags prove to be very important in determining the temporal location of a situation. The reason for using "correct" parse trees in the Chinese Treebank is to factor out noises that are inevitable in the output of an automatic parser and evaluate the contribution of syntactic information in the "ideal" scenario. In a realistic setting, one of course has to use an automatic parser.

The results are presented in Table 1. The overall accuracy is 67.1%, exceeding the baseline of choosing the most frequent tense in the test, which is 62.4%. It is worth noting that the training accuracy is fairly high, 93%, and there is a steep drop-off from the training accuracy to the test accuracy although this is hardly unexpected given the relatively small training set. The high training accuracy nevertheless attests the relevance of the features we have chosen for the classification, which we will look at in greater detail in the next section.

tense	precision	recall	f-score
present	0.51	0.62	0.56
past	0.75	0.81	0.78
future	0.33	0.45	0.38
future-in-past	0.76	0.18	0.30
none	0.86	0.83	0.84
overall	0.93 (train), 0.671 (test)		

Table 1: Experimental results

4 What information is useful?

Our classification algorithm scans the verbs in a sentence one at a time, from left to right. Features are extracted from the context of the verb in the parse tree as well as from previous verbs the tense of which have already been examined. We view features for the classification algorithm as information that contributes to the determination of the temporal location of situations in the absence of morphological markers of tense. The features we used for the classification task can all be extracted from a parse tree and the POS information of a word. They are described below:

- Verb Itself: The character string of the verbs, e.g., 拥有("own"), 是("be"), etc.

- Verb POS: The part-of-speech tag of the verb, as defined in the Chinese Treebank. There are three POS tags for verbs, *VE* for existential verbs such as 有("have, exist"), *VC* for copula verbs like 是("be"), *VA* for stative verbs like 高("tall"), and *VV* for all other verbs.

- Position of verb in compound: If the target verb is part of a verb compound, the position of the compound is used as a feature in combination with the compound type. The possible values for the position are *first* and *last*, and the compound type is one of the six defined in the Chinese Treebank: *VSB*, *VCD*, *VRD*, *VCP*, *VNV*, and *VPT*. An example feature might be "last+VRD".

- Governing verb and its tense: Chinese is an SVO language, and the governing verb, if there is one, is on the left and is higher up in the tree. Since we are scanning verbs in a sentence from left to right, the tense for the governing verb is available at the time we look at the target verb. So we are using the character string of the governing verb as well as its tense as features. In cases where there are multiple levels of embedding and multiple governing verbs, we select the closest governing verb.

- Left ADV: Adverbial modifiers of the target verb are generally on the left side of the verb, therefore we are only extracting adverbs on the left. We first locate the adverbial phrases and then find the head of the adverbial phrase and use character string of the head as feature.

- Left NT: NT is a POS in the Chinese Treebank for nominal expressions that are used as temporal modifiers of a verb. The procedure for extracting the NT modifers is similar to the procedure for finding adverbial modifiers, the only difference being that we are looking for NPs headed by nouns POS-tagged NT.

- Left PP: Like adverbial modifiers, PP modifiers are also generally left modifiers of a verb. If there is a PP modifier, the character string of the head preposition combined with the character string of the head noun of its NP complement is used as a feature, e.g., "在+期间" ("at+period").

- Left LC: Left localizer phrases. Localizers phrases are also called post-positions by some and they function similarly as left PP modifiers. If the target verb has a left localizer phrase modifier and the character string of its head is used as a feature, e.g., 以来("since").

- Left NN: This feature is intended to capture the head of the subject NP. The character string of the head of the NP is used as a feature.

- Aspect marker. Aspect markers are grammaticalizations of aspect and they immediately follow the verb. If the target verb is associated with an aspect marker, the character string of that aspect marker is used as a feature, e.g., "了".

- DER: DER is the POS tag for 得, a character which introduces a resultative construction when following a verb. When it occurs together with the target verb, it is used as a feature.

- Quantifier in object: When there is a quantifier in the NP object for the target verb, its character string is used as a feature.

- Quotation marks: Finally the quotation marks are used as a feature when they are used to quote the clause that contains the target verb.

We performed an ablation evaluation of the features to see how effective each feature type is. Basically, we took out each feature type, retrained the classifier and reran the classifier on the test data. The accuracy without each of the feature types are presented in Table 2. The features are ranked from the most effective to the least effective. Features that lead to the most drop-off when they are taken out of the classification algorithm are considered to be the most effective. As shown in Table 2, the most effective features are the governing verb and its tense, while the least effective features are the quantifiers in the object. Most of the features are lexicalized in that the character strings of words are used as features. When lexicalized features are used, features that appear in the training data do not necessarily appear in the test data and vice versa. This provides a partial explanation of the large discrepancy between the training and test accuracy. In order to reduce this discrepancy, one would have to use a larger training set, or make the features more generalized. Some of these features can in fact be generalized or normalized. For example, a temporal modifier such as the date "1987" can be reduced to something like "before the document creation time", and this is something that we will experiment with in our future work. The training set used here is sufficient to show the efficacy of the features, but to improve the tense classification to a satisfactory level of accuracy, more training data need to be annotated.

Feature	accuracy (w/o)
Governing verb/tense	0.620
verb itself	0.635
Verb POS	0.656
Position verb in compound	0.656
left ADV	0.657
left NT	0.657
Quotation mark	0.657
left PP	0.663
left LC	0.664
Right DER	0.665
Aspect marker	0.665
left NN	0.665
Quantifier in object	0.669
overall	0.671 (test)

Table 2: Feature Performance

Features like adverbial, prepositional, localizer phrase modifiers and temporal noun modifiers provide explicit temporal information that is relevant in determining the temporal location. The role of the governing verb in determining the temporal location of a situation is also easy to understand. As we have shown in Section 2, when the target verb occurs in an embedded clause, its temporal location is necessarily affected by the temporal location of the governing verb of this embedded clause because the temporal location of the former is often defined in relation to that of the latter. Not surprisingly, the governing verb proves to be the most effective feature. Quotation marks in written text change the temporal deixis from the document creation time to the moment of speech of the quoted speaker, and the temporal location in quoted speech does not follow the same patterns as target verbs in embedded clauses. Aspect markers are tied closely to tense, even though the contributions they made are small due to their rare occurrences in text.

The relevance of other features are less obvious. The target verb itself and its POS made the most contribution other than the governing verb. It is important to understand why they are effective or useful at all. In a theoretic work on the temporal interpretation of verbs in languages like Chinese which lacks tense morphology, Smith and Erbaugh (2005) pointed out that there is a default interpretation for bounded and unbounded situations. Specifically, bounded situations are temporally located in the past by default while unbounded situations are located in the future. The default interpretation, by definition, can be overwritten when there is explicit evidence to the contrary. Recast in statistical terms, this means that bounded events have a tendency to be located in the past while unbounded events have a tendency to be located in the present, and this tendency can be quantified in a machine-learning framework. Boundedness has many surface manifestations that can be directly observed, and one of them is whether the verb is stative or dynamic. The target verb itself and its POS tag represents this information. Resultatives in the form of resultative verb compound and the DER construction, quantifiers in the object are other surface reflections of the abstract notion of boundedness. The fact that these features have contributed to the determination of the temporal location of situations to certain extent lends support to Smith's theoretical claim.

5 Related work

Inferring the temporal location is a difficult problem that is not yet very well understood. It has not been studied extensively in the context of Natural Language Processing. Olson et al (2000; 2001) realized the importance of using the aspectual information (both grammatical and lexical aspect) to infer tense in the context of a Chinese-English Machine Translation system. They encoded the aspectual information such as telicity as part of the Lexical Conceptual Structure and use it to heuristically infer tense when generating the English output. This rule-based approach is not very suited for modeling the temporal location information in Chinese. As they themselves noted, aspectual information can only be used as a tendency rather than a deterministic rule. We believe this problem can be better modeled in a machine learning framework where different sources of information, each one being imperfect, can be combined based on their effectiveness to provide a more reasonable overall prediction.

Ye (2007) did approach this problem with machine learning techniques. She used Chinese-English parallel data to manually map the tense information from English to Chinese and trained a Conditional Random Field classifier to make predictions about tense. She used only a limited number of surface cues such as temporal adverbials and aspect markers as features and did not attempt to model the lexical aspect information such as boundedness, which we believe would have helped her system performance. Her data appeared to have a much larger percentage of verb instances that have the past tense and thus her results are mostly incomparable with that of ours.

6 Conclusion and future work

We have defined the automatic inference of the temporal location of situations in Chinese text as a machine learning problem and demonstrated that a lot more information in the form of features contributes to the solution of this challenging problem than previously realized. The accuracy on the held-out test is a significant improvement over the baseline, the proportion of verbs assigned the most frequent tense (the past tense). Although there is a large drop-off from the training accuracy to the test accuracy due to the lexical nature of the features, the high training accuracy does show promise that this challenging problem is solvable with a larger training set, better modeling techniques and more refined features. In the future we will attempt to solve this problem along these lines and work toward a system that can be used in practical applications.

Acknowledgments

We would like to thank Hua Zhong and Kaiyun Chen for their efforts to annotate the data used in our experiments. Without their help this work would of course be impossible.

References

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Mari Olson, David Traum, Carol Vaness Dykema, Amy Weinberg, and Ron Dolan. 2000. Telicity as a cue to temporal and discourse structure in Chinese-English Machine Translation. In *Proceedings of NAACL-ANLP 2000 Workshop on Applied interlinguas: practical applications of interlingual approaches to NLP*, pages 34–41, Seattle Washington.

Mari Olson, David Traum, Carol Vaness Dykema, and Amy Weinberg. 2001. Implicit cues for explicit generation: using telicity as a cue for tense structure in a Chinese to English MT system. In *Proceedings of Machine Translation Summit VIII*, Santiago de Compostela, Spain.

Hans Reichenbach. 1947. *Elements of Symbolic Logic*. The MacMillan Company, New York.

Carlota S. Smith and Mary Erbaugh. 2005. Temporal interpretation in Mandarin Chinese. *Linguistics*, 43(4):713–756.

Nianwen Xue, Fei Xia, Fu dong Chiou, and Martha Palmer. 2005. The Penn Chinese TreeBank: Phrase Structure Annotation of a Large Corpus. *Natural Language Engineering*, 11(2):207–238.

Nianwen Xue, Zhong Hua, and Kai-Yun Chen. 2008. Annotating "tense" in a tenseless language. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, Marrakech, Morocco.

Yang Ye. 2007. *Automatica Tense and Aspect Translation between Chinese and English*. Ph.D. thesis, University of Michigan.

Learning the Scope of Negation in Biomedical Texts

Roser Morante†, Anthony Liekens‡, Walter Daelemans†
CNTS - Language Technology Group†, Applied Molecular Genomics Group‡
University of Antwerp
Prinsstraat 13, B-2000 Antwerpen, Belgium
{Roser.Morante,Anthony.Liekens,Walter.Daelemans}@ua.ac.be

Abstract

In this paper we present a machine learning system that finds the scope of negation in biomedical texts. The system consists of two memory-based engines, one that decides if the tokens in a sentence are negation signals, and another that finds the full scope of these negation signals. Our approach to negation detection differs in two main aspects from existing research on negation. First, we focus on finding the scope of negation signals, instead of determining whether a term is negated or not. Second, we apply supervised machine learning techniques, whereas most existing systems apply rule-based algorithms. As far as we know, this way of approaching the negation scope finding task is novel.

1 Introduction

In this paper we present a machine learning system that finds the scope of negation in biomedical texts. The system consists of two classifiers, one that decides if the tokens in a sentence are negation signals (i.e., words indicating negation), and another that finds the full scope of these negation signals. Finding the scope of a negation signal means determining at sentence level which words in the sentence are affected by the negation. Our approach differs in two main aspects from existing research. First, we focus on finding the scope of negation signals, instead of determining whether a term is negated or not. Second, we apply supervised machine learning techniques, whereas most existing systems apply rule-based algorithms.

Predicting the scope of negation is important in information extraction from text for obvious reasons; instead of simply flagging the sentences containing negation as not suited for extraction (which is currently the best that can be done), correct semantic relations can be extracted when the scope of negation is known, providing a better recall.

Not being able to recognize negation can also hinder automated indexing systems (Mutalik et al., 2001; Rokach et al., 2008). As Mutalik et al. (2001) put it, “to increase the utility of concept indexing of medical documents, it is necessary to record whether the concept has been negated or not”. They highlight the need to detect negations in examples like “no evidence of fracture”, so that an information retrieval system does not return irrelevant reports.

Szarvas et al. (2008) report that 13.45% of the sentences in the abstracts section of the BioScope corpus and 13.76% of the sentences in the full papers section contain negations. A system that does not deal with negation would treat these cases as false positives.

The goals of this research are to model the scope finding task as a classification task similar to the semantic role labeling task, and to test the performance of a memory–based system that finds the scope of negation signals. Memory-based language processing (Daelemans and van den Bosch, 2005) is based on the idea that NLP problems can be solved by reuse of solved examples of the problem in memory, applying similarity-based reasoning on these examples in order to solve new problems. As language processing tasks typically involve many subregularities and (pockets of) exceptions, it has been argued that lazy learning is at an advantage in solving these highly disjunctive learning problems compared to eager learning, as the latter eliminates not only noise but also potentially useful exceptions (Daelemans et al., 1999). Memory-based algorithms have been successfully applied in language processing to a wide range of linguistic tasks, from phonology to semantic analysis, such as semantic role labeling (Morante et al., 2008).

The paper is organised as follows. In Section 2, we summarise related work. In Section 3, we describe the corpus with which the system has been trained. In Section 4, we introduce the task to be performed by the system, which is described in Section 5. The results are presented and discussed in Section 6. Finally, Section 7 puts forward some conclusions.

2 Related work

Negation has been a neglected area in open-domain natural language processing. Most research has been performed in the biomedical domain and has focused on detecting if a medical term is negated or not, whereas in this paper we focus on detecting the full scope of negation signals.

Chapman et al. (2001) developed NegEx, a regular expression based algorithm for determining whether a finding or disease mentioned within narrative medical reports is present or absent. The reported results are 94.51 precision and 77.84 recall.

Mutalik et al. (2001) developed Negfinder, a rule-based system that recognises negated patterns in medical documents. It consists of two tools: a lexical scanner called *lexer* that uses regular expressions to generate a finite state machine, and a parser. The reported results are 95.70 recall and 91.80 precision.

Sanchez-Graillet and Poesio (2007) present an analysis of negated interactions in biological texts and a heuristics-based system that extracts such information. They treat all types of negation: (i) Affixal negation, which is expressed by an affix. (ii) Noun phrase or emphatic negation, expressed syntactically by using a negative determiner (e.g. *no*, *nothing*). (iii) Inherent negation, expressed by words with an inherently negative meaning (e.g. *absent*). (iv) Negation with explicit negative particles (e.g. *no*, *not*). The texts are 50 journal articles. The preliminary results reported range from 54.32 F-score to 76.68, depending on the method applied.

Elkin et al. (2005) describe a rule-based system that assigns to concepts a level of certainty as part of the generation of a dyadic parse tree in two phases: First a preprocessor breaks each sentence into text and operators. Then, a rule based system is used to decide if a concept has been positively, negatively, or uncertainly asserted. The system achieves 97.20 recall and 98.80 precision.

The systems mentioned above are essentially based on lexical information. Huang and Lowe (2007) propose a classification scheme of negations based on syntactic categories and patterns in order to locate negated concepts, regardless of their distance from the negation signal. Their hybrid system that combines regular expression matching with grammatical parsing achieves 92.60 recall and 99.80 precision.

Additionally, Boytcheva et al. (2005) incorporate the treatment of negation in a system, MEHR, that extracts from electronic health records all the information required to generate automatically patient chronicles. According to the authors "the negation treatment module inserts markers in the text for negated phrases and determines scope of negation by using negation rules". However, in the paper there is no description of the rules that are used and it is not explained how the results presented for negation recognition (57% of negations correctly recognised) are evaluated.

The above-mentioned research applies rule-based algorithms to negation finding. Machine learning techniques have been used in some cases. Averbuch et al. (2004) developed an algorithm that uses information gain to learn negative context patterns.

Golding and Chapman (2003) experiment with machine learning techniques to distinguish whether a medical observation is negated by the word *not*. Their corpus contains 207 selected sentences from hospital reports, in which a negation appears. They use Naive Bayes and Decision Trees and achieve a maximum of 90 F-score. According to the authors, their main finding is that "when negation of a UMLS term is triggered with the negation phrase *not*, if the term is preceded by *the* then do not negate".

Goryachev et al. (2006) compare the performance of four different methods of negation detection, two regular expression-based methods and two classification-based methods trained on 1745 discharge reports. They show that the regular expression-based methods have better agreement with humans and better accuracy than the classification methods. Like in most of the mentioned work, the task consists in determining if a medical term is negated.

Rokach et al. (2008) present a new pattern-based algorithm for identifying context in free-text medical narratives.The originality of the algorithm lies in that it automatically learns patterns similar to the manually written patterns for negation detection.

Apart from work on determining whether a term is negated or not, we are not aware of research that has focused on learning the full scope of negation signals inside or outside biomedical natural language processing. The research presented in this paper provides a new approach to the treatment of negation scope in natural language processing.

3 Corpus

The corpus used is a part of the BioScope corpus (Szarvas et al., 2008)[1], a freely available resource that consists of medical and biological texts. Every sentence is annotated with information about negation and speculation that indicates the boundaries of the scope and the keywords, as shown in (1).

(1) PMA treatment, and <xcope id="X1.4.1"><cue type="negation" ref="X1.4.1">not<cue> retinoic acid treatment of the U937 cells</xcope> acts in inducing NF-KB expression in the nuclei.

A first characteristic of the annotation of scope in the BioScope corpus is that all sentences that assert the non-existence or uncertainty of something are annotated, in contrast to other corpora where only sentences of interest in the domain are annotated. A second characteristic is that the annotation is extended to the biggest syntactic unit possible so that scopes have the maximal length. In (2) below, negation signal *no* scopes over *primary impairment of glucocorticoid metabolism* instead of scoping only over *primary*.

(2) There is [no] primary impairment of glucocorticoid metabolism in the asthmatics.

[1]Web page: `www.inf.u-szeged.hu/rgai/bioscope`.

The part used in our experiments are the biological paper abstracts from the GENIA corpus (Collier et al., 1999). This part consists of 11,872 sentences in 1,273 abstracts. We automatically discarded five sentences due to annotation errors. The total number of words used is 313,222, 1,739 of which are negation signals that belong to the different types described in (Sanchez-Graillet and Poesio, 2007).

We processed the texts with the GENIA tagger (Tsuruoka and Tsujii, 2005; Tsuruoka et al., 2005), a bidirectional inference based tagger that analyzes English sentences and outputs the base forms, part-of-speech tags, chunk tags, and named entity tags in a tab-separated format[2]. Additionally, we converted the annotation about scope of negation into a token-per-token representation.

Table 1 shows an example sentence of the corpus that results from converting and processing the BioScope representation. Following the standard format of the CoNLL Shared Task 2006 (Buchholz and Marsi, 2006), sentences are separated by a blank line and fields are separated by a single tab character. A sentence consists of tokens, each one starting on a new line. A token consists of the following 10 fields:

1. ABSTRACT ID: number of the GENIA abstract.
2. SENTENCE ID: sentence counter starting at 1 for each new abstract.
3. TOKEN ID: token counter, starting at 1 for each new sentence.
4. FORM: word form or punctuation symbol.
5. LEMMA: lemma of word form.
6. POS TAG: Penn Treebank part-of-speech tags described in (Santorini, 1990).
7. CHUNK TAG: IOB (Inside, Outside, Begin) tags produced by the GENIA tagger that indicate if a token is inside a certain chunk, outside, or at the beginning.
8. NE TAG: IOB named entity tags produced by the GENIA tagger that indicate if a token is in-

[2]The accuracy of the tagger might be inflated due to the fact that it was trained on the GENIA corpus.

ABSTR ID	SNT ID	TOK ID	FORM	LEMMA	POS TAG	CHUNK TAG	NE TAG	NEG SGN	NEG SCOPE
10415075	07	1	NF-kappa	NF-kappa	NN	B-NP	B-protein	_	I-NEG O-NEG
10415075	07	2	B	B	NN	I-NP	I-protein	_	I-NEG O-NEG
10415075	07	3	binding	binding	NN	I-NP	O	_	I-NEG O-NEG
10415075	07	4	activity	activity	NN	I-NP	O	_	I-NEG O-NEG
10415075	07	5	was	be	VBD	B-VP	O	_	I-NEG O-NEG
10415075	07	6	absent	absent	JJ	B-ADJP	O	NEG	I-NEG O-NEG
10415075	07	7	in	in	IN	B-PP	O	_	I-NEG O-NEG
10415075	07	8	several	several	JJ	B-NP	O	_	I-NEG O-NEG
10415075	07	9	SLE	SLE	NN	I-NP	O	_	I-NEG O-NEG
10415075	07	10	patients	patient	NNS	I-NP	O	_	I-NEG O-NEG
10415075	07	11	who	who	WP	B-NP	O	_	I-NEG O-NEG
10415075	07	12	were	be	VBD	B-VP	O	_	I-NEG O-NEG
10415075	07	13	not	not	RB	I-VP	O	NEG	I-NEG I-NEG
10415075	07	14	receiving	receive	VBG	I-VP	O	_	I-NEG I-NEG
10415075	07	15	any	any	DT	B-NP	O	_	I-NEG I-NEG
10415075	07	16	medication	medication	NN	I-NP	O	_	I-NEG I-NEG
10415075	07	17	,	,	,	O	O	_	I-NEG I-NEG
10415075	07	18	including	include	VBG	B-PP	O	_	I-NEG I-NEG
10415075	07	19	corticosteroids	corticosteroid	NNS	B-NP	O	_	I-NEG I-NEG
10415075	07	20	.	.	.	O	O	_	O-NEG O-NEG

Table 1: Example sentence of the BioScope corpus converted into columns format.

side a certain named entity, outside, or at the beginning.

9. NEG SIGNAL: tokens that are negation signals are marked as NEG. Negation signals in the BioScope corpus are not always single words, like the signal *could not*. After the tagging process the signal *cannot* becomes also multiword because the tagger splits it in two words. In these cases we assign the NEG mark to *not*.

10. NEG SCOPE: IO tags that indicate if a token is inside the negation scope (I-NEG), or outside (O-NEG). These tags have been obtained by converting the xml files of BioScope. Each token can have one or more NEG SCOPE tags, depending on the number of negation signals in the sentence.

4 Task description

We approach the scope finding task as a classification task that consists of classifying the tokens of a sentence as being a negation signal or not, and as being inside or outside the scope of the negation signal(s). This happens as many times as there are negation signals in the sentence. Our conception of the task is inspired by Ramshaw and Marcus' representation of text chunking as a tagging problem (Ramshaw and Marcus, 1995) .

The information that can be used to train the system appears in columns 1 to 8 of Table 1. The information to be predicted by the system is contained in columns 9 and 10.

As far as we know, approaching the negation scope finding task as a token per token classification task is novel, whereas at the same time it conforms to the well established standards of the recent CoNLL Shared Tasks[3] on dependency parsing (Buchholz and Marsi, 2006; Nivre et al., 2007) and semantic role labeling (Surdeanu et al., 2008). By setting up the task in this way we show that the negation scope finding task can be modelled in a way similar to semantic role labeling, and by conforming to existing standards we show that learning the scope of negation can be integrated in a joint learning task with dependency parsing and semantic role labeling.

[3]Web page of CoNLL: `http://www.ifarm.nl/signll/conll/`.

5 System description

In order to solve the task, we apply supervised machine learning techniques. We build a memory-based scope finder, that tackles the task in two phases. In the first phase a classifier predicts if a token is a negation signal, and in the second phase another classifier predicts if a token is inside the scope of each of the negation signals. Additionally, the output of the second classifier is postprocessed with an algorithm that converts non-consecutive blocks of scope into consecutive, as explained in Section 5.3.

As for the first and second phases, we use a memory–based classifier as implemented in TiMBL (version 6.1.2) (Daelemans et al., 2007), a supervised inductive algorithm for learning classification tasks based on the k-nearest neighbor classification rule (Cover and Hart, 1967). Similarity is defined by computing (weighted) overlap of the feature values of a test instance and training instances. The metric combines a per-feature value distance metric (Cost and Salzberg, 1993) with gain ratio (Quinlan, 1993) based global feature weights that account for relative differences in discriminative power of the features.

5.1 Negation signal finding

In this phase, a classifier predicts whether a token is a negation signal or not. The memory-based classifier was parameterised by using overlap as the similarity metric, gain ratio for feature weighting, and using 7 k-nearest neighbors. All neighbors have equal weight when voting for a class. The instances represent all tokens in the corpus and they have the following features:

- Of the token: Form, lemma, part of speech, and chunk IOB tag.
- Of the token context: Form, POS, and IOB tag of the three previous and three next tokens.

5.2 Scope finding

In the first step of this phase, a classifier predicts whether a token is in the scope of each of the negation signals of a sentence. A pair of a negation signal and a token from the sentence represents an instance. This means that all tokens in a sentence are paired with all negation signals that occur in the sentence. For example, token *NF-kappa* in Table 1 will be represented in two instances as shown in (3). An instance represents the pair [NF–KAPPA, absent] and another one represents the pair [NF–KAPPA, not].

(3) NF-kappa absent [features] I-NEG
NF-kappa not [features] O-NEG

Negation signals are those that have been classified as such in the previous phase. Only sentences that have negation signals are selected for this phase.

The memory–based algorithm was parameterised in this case by using overlap as the similarity metric, gain ratio for feature weighting, using 7 k-nearest neighbors, and weighting the class vote of neighbors as a function of their inverse linear distance.

The features of the scope finding classifier are:

- Of the negation signal: Form, POS, chunk IOB tag, type of chunk (NP, VP, ...), and form, POS, chunk IOB tag, type of chunk, and named entity of the 3 previous and 3 next tokens.
- Of the paired token: form, POS, chunk IOB tag, type of chunk, named entity, and form, POS, chunk IOB tag, type of chunk, and named entity type of the 3 previous and 3 next tokens.
- Of the tokens between the negation signal and the token in focus: Chain of POS types, distance in number of tokens, and chain of chunk IOB tags.
- Others: A binary feature indicating whether the token and the negation signal are in the same chunk, and location of the token relative to the negation signal (pre, post, same).

5.3 Post-processing

Negation signals in the BioScope corpus always have one consecutive block of scope tokens, including the signal token itself. However, the scope finding classifier can make predictions that result in non-consecutive blocks of scope tokens: we observed that 54% of scope blocks predicted by the system given gold standard negation signals are non–consecutive. This is why in the second step of the scope finding phase, we apply a post-processing algorithm in order to increase the number of fully correct scopes. A scope is fully correct if all tokens in a sentence have been assigned their correct class label for a given negation signal. Post-processing ensures that the resulting scope is one consecutive block of tokens.

In the BioScope corpus negation signals are inside of their scope. The post-processing algorithm that we apply first checks if the negation signal is in its scope. If the signal is out, the algorithm overwrites the predicted scope in order to include the signal in its scope.

Given the position of the signal in the sentence, the algorithm locates the starting and ending tokens of the consecutive block of predicted scope tokens that surrounds the signal. Other blocks of predicted scope tokens may have been predicted outside of this block, but they are separated from the current block, which contains the signal, by tokens that have been predicted not to be in the scope of the negation, as in Figure 1.

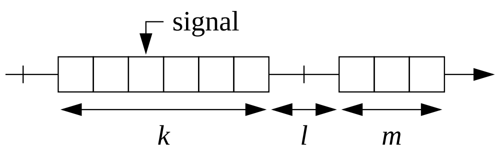


Figure 1: Non-consecutive blocks of scope tokens. For a signal, two blocks of $k = 6$ and $m = 3$ tokens are predicted to be the scope of the signal token, but they are separated by $l = 2$ tokens that are predicted to be out of scope.

The post-processing algorithm decides whether the detached blocks should be connected as one consecutive block of scope tokens, or whether the detached block of scope tokens should be discarded from the scope. Dependent on this decision, either the classification of the separated blocks, or the separating non-scope tokens are considered noisy, and their classification is updated to produce one consecutive block of scope tokens for each signal. This check is performed iteratively for all detached blocks of scope tokens.

As in Figure 1, consider a sentence where the negation signal is in one block K of predicted scope of length k tokens and another block M of m consecutive tokens that is predicted as scope but is separated from the latter scope block by l out-of-scope tokens.

If non-consecutive blocks are near each other, i.e., if l is sufficiently small in comparison with k and m, then the intermediate tokens that have been predicted out of scope could be considered as noise and converted into scope tokens. In contrast, if there are too many intermediate tokens that separate the two blocks of scope tokens, then the additional block of scope is probably wrongly classified.

Following this logic, if $l < \alpha(k + m)$, with a specifically chosen α, the intermediate out-of-scope tokens are re-classified as scope tokens, and the separated blocks are connected to form one bigger block containing the negation signal. Otherwise, the loose block of scope is re-classified to be out of scope. When the main scope is extended, and more blocks are found that are separated from the main scope block, the algorithm reiterates this procedure until one consecutive block of scope tokens has been found.

Our implementation first looks for separated blocks from right to left, and then from left to right. Dependent on whether blocks need to be added before or after the main scope block, we have observed in preliminary tests that $\alpha = 0.2$ for extending the main scope block from right to left, and $\alpha = 0.3$ for extending the block from left to right into the sentence provide the best results. Algorithm 1 details the above procedure in pseudo code.

Algorithm 1 Post-processing

```
$K \leftarrow$ scope block that contains signal
while $M \leftarrow$ nearest separated scope block do
    $L \leftarrow$ non-scope block between $K$ and $M$
    if $|L| < \alpha(|K| + |M|)$ then
        include $L$ in scope
    else
        exclude $M$ from scope
    end if
    $K \leftarrow$ scope block that contains signal
end while
```

6 Results

The results have been obtained by performing 10-fold cross validation experiments. The evaluation is made using the precision and recall measures (Van Rijsbergen, 1979), and their harmonic mean, F-Measure. We calculate micro F1.

In the negation finding task, a negation token is correctly classified if it has been assigned a NEG class. In the scope finding task, a token is correctly classified if all the IO tag(s) that it has been assigned are correct. This means that when there is more than one negation signal in the sentence, the token has to be correctly assigned an IO tag for as many negation signals as there are. For example, token NF-kappa from Table 1 reproduced in (4) will not be correct if it is assigned classes I-NEG I-NEG or O-NEG I-NEG.

(4) 10415075 07 1 NF-kappa NF-kappa NN B-NP B-protein _ I-NEG O-NEG

Additionally, we evaluated the percentage of fully correct scopes (PCS).

6.1 Negation signal finding

We calculate two baselines for negation signal finding. Baseline 1 (B1) is calculated by assigning the NEG class to all the tokens that had *no* or *not* as lemma, which account for 72.80% of the negation signals. The F1 of the baseline is 80.66. Baseline 2 (B2) is calculated by assigning the NEG class to all the tokens that had *no*, *not*, *lack*, *neither*, *unable*, *without*, *fail*, *absence*, or *nor* as lemma. These lemmas account for 85.85 % of the negation signals.

Baseline	Total	Prec.	Recall	F1
B1	1739	90.42	72.80	80.66
B2	1739	89.77	93.38	91.54

Table 2: Baselines of the negation finding system.

Table 3 shows the overall results of the negation signal finding system and the results per negation signal. With F1 94.40, it outperforms Baseline 2 by 2.86 points. Precision and recall are very similar. Scores show a clear unbalance between different negation signals. Those with the lowest frequencies get lower scores than those with the highest frequencies. Probably, this could be avoided by training the system with a bigger corpus.

However, a bigger corpus would not help solve all the errors because some of them are caused by inconsistency in the annotation. For example, *absence* is annotated as a negation signal in 57 cases, whereas in 22 cases it is not annotated as such, although in all cases it is used as a negation signal. Example 5 (a) shows one of the 22 cases of *absence* that has not been annotated, and Example 5 (b) shows one of the 57 cases of *absence* annotated as a negation signal. Also *fail* is not annotated as a negation signal in 13 cases where it should.

Neg signals	Total	Prec.	Recall	F1
lack (v)	55	100.00	100.00	100.00
neither (con)	34	100.00	100.00	100.00
lack (n)	33	100.00	100.00	100.00
unable	30	100.00	100.00	100.00
neither (det)	8	100.00	100.00	100.00
no (adv)	5	100.00	100.00	100.00
without	83	100.00	98.79	99.39
nor	44	100.00	100.00	98.89
rather	19	95.00	100.00	97.43
not	1057	96.15	96.97	96.56
no (det)	204	95.63	96.56	96.09
none	7	85.71	85.71	85.71
fail	57	79.36	87.71	83.33
miss	2	66.66	100.00	80.00
absence	57	67.64	80.70	73.60
failure	8	45.54	62.50	52.63
could	6	66.66	33.33	44.44
absent	13	42.85	23.07	30.00
with	6	0.00	0.00	0.00
either	2	0.00	0.00	0.00
instead	2	0.00	0.00	0.00
never	2	0.00	0.00	0.00
impossible	1	0.00	0.00	0.00
lacking	1	0.00	0.00	0.00
loss	1	0.00	0.00	0.00
negative	1	0.00	0.00	0.00
or	1	0.00	0.00	0.00
Overall	1739	94.21	94.59	94.40

Table 3: F scores of the negation finding classifier.

(5) (a) Retroviral induction of TIMP-1 not only resulted in cell survival but also in continued DNA synthesis for up to 5 d in the **absence** of serum, while controls underwent apoptosis.

(b) A significant proportion of transcripts appear to terminate prematurely in the <xcope id= X654.8.1 ><cue type= negation ref= X654.8.1 > **absence** </cue> of transactivators </xcope>.

Other negation signals are arbitrarily annotated. *Failure* is annotated as a negation signal in 8 cases where it is followed by a preposition, like in Example 6 (a), and it is not annotated as such in 26 cases, like Example 6 (b), where it is modified by an adjective.

(6) (a) ... the <xcope id= X970.8.2> <cue type= negation ref= X970.8.2>**failure**</cue> of eTh1 cells to produce IL-4 in response to an antigen </xcope> is due, at least partially, to a <xcope id= X970.8.1> < cue type= negation ref= X970.8.1> **failure**</cue> to induce high-level transcription of the IL-4 gene by NFAT </xcope></xcope>.

(b) Positive-pressure mechanical ventilation supports gas exchange in patients with respiratory **failure** but is also responsible for significant lung injury.

The errors in detecting *with* as a negation signal are caused by the fact that it is embedded in the expression *with the exception of*, which occurs 6 times in contrast with the 5265 occurrences of *with*. *Could* appears as a negation signal because the tagger does not assign to it the lemma *can*, but *could*, causing the wrong assignment of the tag NEG to *not*, instead of *could* when the negation cue in BioScope is *could not*.

6.2 Scope finding

We provide the results of the classifier and the results of applying the postprocessing algorithm to the output of the classifier.

Table 4 shows results for two versions of the scope finding classifier, one based on gold standard negation signals (GS NEG), and another (PR NEG) based on negation signals predicted by the classifier described in the previous section.

	Prec.	**Recall**	**F1**	**PCS**
GS NEG	86.03	85.53	85.78	39.39
PR NEG	79.83	77.42	78.60	36.31

Table 4: Results of the scope finding classifier with gold-standard (GS NEG) and with predicted negation signals (PR NEG).

The F1 of PR NEG is 7.18 points lower than the F1 of GS NEG, which is an expected effect due to the performance of classifier that finds negation signals. Precision and recall of GS NEG are very balanced, whereas PR NEG has a lower recall than precision. These measures are the result of a token per token evaluation, which does not guarantee that the complete sequence of scope is correct. This is reflected in the low percentage of fully correct scopes of both versions of the classifier.

In Table 5, we present the results of the system after applying the postprocessing algorithm. The most remarkable result is the 29.60 and 21.58 error reduction in the percentage of fully correct scopes of GS NEG and PR NEG respectively, which shows that the algorithm is efficient. Also interesting is the increase in F1 of GS NEG and PR NEG.

	Prec.	**Recall**	**F1**	**PCS**
GS NEG	88.63	88.17	88.40	57.33
PR NEG	80.70	81.29	80.99	50.05

Table 5: Results of the system with gold-standard (GS NEG) and with predicted negation signals (PR NEG) after applying the postprocessing algorithm.

Table 6 shows detailed results of the system based on predicted negation signals after applying the postprocessing algorithm. Classes O-NEG and I-NEG are among the most frequent and get high scores. Classes composed only of O-NEG tags are easier to predict.

Scope tags	**Total**	**Prec.**	**Recall**	**F1**
O-NEG	29590	86.78	84.75	85.75
O-NEG O-NEG O-NEG	46	100.00	63.04	77.33
I-NEG	12990	73.41	80.72	76.89
O-NEG O-NEG	2848	84.11	68.43	75.46
I-NEG I-NEG O-NEG	69	62.92	81.15	70.88
I-NEG I-NEG	684	57.30	65.93	61.31
I-NEG O-NEG O-NEG	20	50.00	75.00	60.00
O-NEG I-NEG	791	72.13	50.06	59.10
I-NEG O-NEG	992	45.32	67.94	54.37
O-NEG I-NEG I-NEG	39	100.00	20.51	34.04
I-NEG I-NEG I-NEG	22	26.66	36.36	30.76
O-NEG O-NEG I-NEG	14	0.00	0.00	0.00
Overall	48105	80.70	81.29	80.99

Table 6: F scores of the system per scope class after applying the postprocessing algorithm.

Table 7 shows information about the percentage of correct scopes per negation signal after applying the algorithm to PR-NEG. A clear example of an incorrect prediction is the occurrence of *box* in the list. The signal with the highest percentage of PCS is *without*, followed by *no* (determiner), *rather* and *not*, which are above 50%. It would be interesting to investigate how the syntactic properties of the negation signals are related to the percentage of correct scopes, and how does the algorithm perform depending on the type of signal.

Neg signals	Total	Correct	PCS
without	82	56	68.29
no (det)	206	133	64.56
rather	20	11	55.00
not	1066	556	52.15
neither (det)	8	4	50.00
none	7	3	42.85
neither (conj)	34	16	47.05
no (adv)	5	2	40.00
fail	63	23	36.50
missing	3	1	33.33
absence	68	22	32.35
lack (v.)	54	17	31.48
absent	7	2	28.57
lack (n.)	33	9	27.27
nor	43	11	25.58
unable	30	8	26.66
failure	11	0	0.00
could	3	0	0.00
negative	1	0	0.00
never	1	0	0.00
box	1	0	0.00
Overall	1746	874	50.05

Table 7: Information about Percentage of Correct Scopes (PCS) per negation signal in PR-NEG.

7 Conclusions

Given the fact that a significant portion of biomedical text is negated, recognising negated instances is important in NLP applications. In this paper we have presented a machine learning system that finds the scope of negation in biomedical texts. The system consists of two memory-based classifiers, one that decides if the tokens in a sentence are negation signals, and another that finds the full scope of the negation signals.

The first classifier achieves 94.40 F1, and the second 80.99. However, the evaluation in terms of correct scopes shows the weakness of the system. This is why a postprocessing algorithm is applied. The algorithm achieves an error reduction of 21.58, with 50.05 % of fully correct scopes in the system based on predicted negation signals.

These results suggest that unsupervised machine learning algorithms are suited for tackling the task, as it was expected from results obtained in other natural language processing tasks. However, results also suggest that there is room for improvement. A first improvement would consist in predicting the scope chunk per chunk instead of token per token, because most negation scope boundaries coincide with boundaries of chunks.

We have highlighted the fact that our approach to negation detection focuses on finding the scope of negation signals, instead of determining whether a term is negated or not, and on applying supervised machine learning techniques. As far as we know, this approach is novel. Unfortunately, there are no previous comparable approaches to measure the quality of our results.

Additionally, we have shown that negation finding can be modelled as a classification task in a way similar to other linguistic tasks like semantic role labeling. In our model, tokens of a sentence are classified as being a negation signal or not, and as being inside or outside the scope of the negation signal(s). This representation would allow to integrate the task with other semantic tasks and exploring the interaction between different types of knowledge in a joint learning setting.

Further research is possible in several directions. In the first place, other machine learning algorithms could be integrated in the system in order to optimise performance. Secondly, the system should be tested in different types of biomedical texts, like full papers or medical reports to check its robustness. Finally, the postprocessing algorithm could be improved by using more sophisticated sequence classification techniques (Dietterich, 2002) .

Acknowledgments

Our work was made possible through financial support from the University of Antwerp (GOA project BIOGRAPH). We are thankful to three anonymous reviewers for their valuable comments and suggestions.

References

M. Averbuch, T. Karson, B. Ben-Ami, O. Maimon, and L. Rokach. 2004. Context-sensitive medical information retrieval. In *Proc. of the 11th World Congress on Medical Informatics (MEDINFO-2004)*, pages 1–8, San Francisco, CA. IOS Press.

S. Boytcheva, A. Strupchanska, E. Paskaleva, and D. Tcharaktchiev. 2005. Some aspects of negation processing in electronic health records. In *Proc. of International Workshop Language and Speech Infrastructure for Information Access in the Balkan Countries*, pages 1–8, Borovets, Bulgaria.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of the X CoNLL Shared Task*, New York. SIGNLL.

W. W. Chapman, W. Bridewell, P. Hanbury, G. F. Cooper, and B.G. Buchanan. 2001. A simple algorithm for identifying negated findings and diseases in discharge summaries. *J Biomed Inform*, 34:301–310.

N. Collier, H.S. Park, N. Ogata, Y. Tateisi, C. Nobata, T. Sekimizu, H. Imai, and J. Tsujii. 1999. The GENIA project: corpus-based knowledge acquisition and information extraction from genome research papers. In *Proceedings of EACL-99*.

S. Cost and S. Salzberg. 1993. A weighted nearest neighbour algorithm for learning with symbolic features. *Machine learning*, 10:57–78.

T. M. Cover and P. E. Hart. 1967. Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, 13:21–27.

W. Daelemans and A. van den Bosch. 2005. *Memory-based language processing*. Cambridge University Press, Cambridge, UK.

W. Daelemans, A. Van den Bosch, and J. Zavrel. 1999. Forgetting exceptions is harmful in language learning. *Machine Learning, Special issue on Natural Language Learning*, 34:11–41.

W. Daelemans, J. Zavrel, K. Van der Sloot, and A. Van den Bosch. 2007. TiMBL: Tilburg memory based learner, version 6.1, reference guide. Technical Report Series 07-07, ILK, Tilburg, The Netherlands.

T. G. Dietterich. 2002. Machine learning for sequential data: A review. In *Lecture Notes in Computer Science 2396*, pages 15–30, London. Springer Verlag.

P. L. Elkin, S. H. Brown, B. A. Bauer, C.S. Husser, W. Carruth, L.R. Bergstrom, and D. L. Wahner-Roedler. 2005. A controlled trial of automated classification of negation from clinical notes. *BMC Medical Informatics and Decision Making*, 5(13).

l. M. Goldin and W.W. Chapman. 2003. Learning to detect negation with 'Not' in medical texts. In *Proceedings of ACM-SIGIR 2003*.

S. Goryachev, M. Sordo, Q.T. Zeng, and L. Ngo. 2006. Implementation and evaluation of four different methods of negation detection. Technical report, DSG.

Y. Huang and H.J. Lowe. 2007. A novel hybrid approach to automated negation detection in clinical radiology reports. *J Am Med Inform Assoc*, 14(3):304–311.

R. Morante, W. Daelemans, and V. Van Asch. 2008. A combined memory-based semantic role labeler of english. In *Proc. of the CoNLL 2008*, pages 208–212, Manchester, UK.

A.G. Mutalik, A. Deshpande, and P.M. Nadkarni. 2001. Use of general-purpose negation detection to augment concept indexing of medical documents. a quantitative study using the UMLS. *J Am Med Inform Assoc*, 8(6):598–609.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL-2007 shared task on dependency parsing. In *Proc. of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, Prague.

J.R. Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

L. Ramshaw and M. Marcus. 1995. Text chunking using transformation-based learning. In *Proc. of ACL Third Workshop on Very Large Corpora*, pages 82–94, Cambridge, MA. ACL.

L. Rokach, R.Romano, and O. Maimon. 2008. Negation recognition in medical narrative reports. *Information Retrieval Online.*

O. Sanchez-Graillet and M. Poesio. 2007. Negation of protein-protein interactions: analysis and extraction. *Bioinformatics*, 23(13):424–432.

B. Santorini. 1990. Part-of-speech tagging guidelines for the penn treebank project. Technical report MS-CIS-90-47, Department of Computer and Information Science, University of Pennsylvania.

M. Surdeanu, R. Johansson, A. Meyers, Ll. Màrquez, and J. Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proc. of CoNLL 2008*, pages 159–177, Manchester, UK.

G. Szarvas, V. Vincze, R. Farkas, and J. Csirik. 2008. The BioScope corpus: annotation for negation, uncertainty and their scopein biomedical texts. In *Proc. of BioNLP 2008*, pages 38–45, Columbus, Ohio, USA. ACL.

Y. Tsuruoka and J. Tsujii. 2005. Bidirectional inference with the easiest-first strategy for tagging sequence data. In *Proc. of HLT/EMNLP 2005*, pages 467–474.

Y. Tsuruoka, Y. Tateishi, J. Kim, T. Ohta, J. McNaught, S. Ananiadou, and J. Tsujii, 2005. *Advances in Informatics - 10th Panhellenic Conference on Informatics*, volume 3746 of *Lecture Notes in Computer Science*, chapter Part-of-Speech Tagger for Biomedical Text, Advances in Informatics, pages 382–392. Springer, Berlin/Heidelberg.

C.J. Van Rijsbergen. 1979. *Information Retrieval*. Butterworths, London.

Lattice-based Minimum Error Rate Training for Statistical Machine Translation

Wolfgang Macherey **Franz Josef Och** **Ignacio Thayer** **Jakob Uszkoreit**

Google Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043, USA
{wmach,och,thayer,uszkoreit}@google.com

Abstract

Minimum Error Rate Training (MERT) is an effective means to estimate the feature function weights of a linear model such that an automated evaluation criterion for measuring system performance can directly be optimized in training. To accomplish this, the training procedure determines for each feature function its exact error surface on a given set of candidate translations. The feature function weights are then adjusted by traversing the error surface combined over all sentences and picking those values for which the resulting error count reaches a minimum. Typically, candidates in MERT are represented as N-best lists which contain the N most probable translation hypotheses produced by a decoder. In this paper, we present a novel algorithm that allows for efficiently constructing and representing the exact error surface of *all* translations that are encoded in a phrase lattice. Compared to N-best MERT, the number of candidate translations thus taken into account increases by several orders of magnitudes. The proposed method is used to train the feature function weights of a phrase-based statistical machine translation system. Experiments conducted on the NIST 2008 translation tasks show significant runtime improvements and moderate BLEU score gains over N-best MERT.

1 Introduction

Many statistical methods in natural language processing aim at minimizing the probability of sentence errors. In practice, however, system quality is often measured based on error metrics that assign non-uniform costs to classification errors and thus go far beyond counting the number of wrong decisions. Examples are the mean average precision for ranked retrieval, the F-measure for parsing, and the BLEU score for *statistical machine translation* (SMT). A class of training criteria that provides a tighter connection between the decision rule and the final error metric is known as *Minimum Error Rate Training* (MERT) and has been suggested for SMT in (Och, 2003).

MERT aims at estimating the model parameters such that the decision under the zero-one loss function maximizes some end-to-end performance measure on a development corpus. In combination with log-linear models, the training procedure allows for a direct optimization of the unsmoothed error count. The criterion can be derived from Bayes' decision rule as follows: Let $\mathbf{f} = f_1, ..., f_J$ denote a source sentence ('French') which is to be translated into a target sentence ('English') $\mathbf{e} = e_1, ..., e_I$. Under the zero-one loss function, the translation which maximizes the *a posteriori* probability is chosen:

$$\hat{\mathbf{e}} = \arg\max_{\mathbf{e}} \big\{ \Pr(\mathbf{e}|\mathbf{f}) \big\} \tag{1}$$

Since the true posterior distribution is unknown, $\Pr(\mathbf{e}|\mathbf{f})$ is modeled via a log-linear translation model which combines some feature functions $h_m(\mathbf{e}, \mathbf{f})$ with feature function weights λ_m, $m = 1, ..., M$:

$$\begin{aligned} \Pr(\mathbf{e}|\mathbf{f}) &= p_{\lambda_1^M}(\mathbf{e}|\mathbf{f}) \\ &= \frac{\exp\big[\sum_{m=1}^M \lambda_m h_m(\mathbf{e}, \mathbf{f})\big]}{\sum_{\mathbf{e}'} \exp\big[\sum_{m=1}^M \lambda_m h_m(\mathbf{e}', \mathbf{f})\big]} \end{aligned} \tag{2}$$

The feature function weights are the parameters of the model, and the objective of the MERT criterion is to find a parameter set λ_1^M that minimizes the error count on a representative set of training sentences. More precisely, let $\mathbf{f}_1^S$ denote the source sentences of a training corpus with given reference translations

$\mathbf{r}_1^S$, and let $\mathbf{C}_s = \{\mathbf{e}_{s,1}, ..., \mathbf{e}_{s,K}\}$ denote a set of K candidate translations. Assuming that the corpus-based error count for some translations $\mathbf{e}_1^S$ is additively decomposable into the error counts of the individual sentences, i.e., $E(\mathbf{r}_1^S, \mathbf{e}_1^S) = \sum_{s=1}^S E(\mathbf{r}_s, \mathbf{e}_s)$, the MERT criterion is given as:

$$\hat{\lambda}_1^M = \arg\min_{\lambda_1^M} \left\{ \sum_{s=1}^S E\big(\mathbf{r}_s, \hat{\mathbf{e}}(\mathbf{f}_s; \lambda_1^M)\big) \right\} \quad (3)$$

$$= \arg\min_{\lambda_1^M} \left\{ \sum_{s=1}^S \sum_{k=1}^K E(\mathbf{r}_s, \mathbf{e}_{s,k}) \delta\big(\hat{\mathbf{e}}(\mathbf{f}_s; \lambda_1^M), \mathbf{e}_{s,k}\big) \right\}$$

with

$$\hat{\mathbf{e}}(\mathbf{f}_s; \lambda_1^M) = \arg\max_{\mathbf{e}} \left\{ \sum_{m=1}^M \lambda_m h_m(\mathbf{e}, \mathbf{f}_s) \right\} \quad (4)$$

In (Och, 2003), it was shown that linear models can effectively be trained under the MERT criterion using a special line optimization algorithm. This line optimization determines for each feature function h_m and sentence $\mathbf{f}_s$ the exact error surface on a set of candidate translations $\mathbf{C}_s$. The feature function weights are then adjusted by traversing the error surface combined over all sentences in the training corpus and moving the weights to a point where the resulting error reaches a minimum.

Candidate translations in MERT are typically represented as N-best lists which contain the N most probable translation hypotheses. A downside of this approach is, however, that N-best lists can only capture a very small fraction of the search space. As a consequence, the line optimization algorithm needs to repeatedly translate the development corpus and enlarge the candidate repositories with newly found hypotheses in order to avoid overfitting on $\mathbf{C}_s$ and preventing the optimization procedure from stopping in a poor local optimum.

In this paper, we present a novel algorithm that allows for efficiently constructing and representing the unsmoothed error surface for *all* translations that are encoded in a phrase lattice. The number of candidate translations thus taken into account increases by several orders of magnitudes compared to N-best MERT. Lattice MERT is shown to yield significantly faster convergence rates while it explores a much larger space of candidate translations which is exponential in the lattice size. Despite this vast search space, we show that the suggested algorithm is always efficient in both running time and memory.

The remainder of this paper is organized as follows. Section 2 briefly reviews N-best MERT and introduces some basic concepts that are used in order to develop the line optimization algorithm for phrase lattices in Section 3. Section 4 presents an upper bound on the complexity of the unsmoothed error surface for the translation hypotheses represented in a phrase lattice. This upper bound is used to prove the space and runtime efficiency of the suggested algorithm. Section 5 lists some best practices for MERT. Section 6 discusses related work. Section 7 reports on experiments conducted on the NIST 2008 translation tasks. The paper concludes with a summary in Section 8.

2 Minimum Error Rate Training on N-best Lists

The goal of MERT is to find a weights set that minimizes the unsmoothed error count on a representative training corpus (cf. Eq. (3)). This can be accomplished through a sequence of line minimizations along some vector directions $\{d_1^M\}$. Starting from an initial point λ_1^M, computing the most probable sentence hypothesis out of a set of K candidate translations $\mathbf{C}_s = \{\mathbf{e}_1, ..., \mathbf{e}_K\}$ along the line $\lambda_1^M + \gamma \cdot d_1^M$ results in the following optimization problem (Och, 2003):

$$\hat{\mathbf{e}}(\mathbf{f}_s; \gamma) = \arg\max_{\mathbf{e} \in \mathbf{C}_s} \left\{ (\lambda_1^M + \gamma \cdot d_1^M)^\top \cdot h_1^M(\mathbf{e}, \mathbf{f}_s) \right\}$$

$$= \arg\max_{\mathbf{e} \in \mathbf{C}_s} \Bigg\{ \underbrace{\sum_m \lambda_m h_m(\mathbf{e}, \mathbf{f}_s)}_{=a(\mathbf{e},\mathbf{f}_s)} + \gamma \cdot \underbrace{\sum_m d_m h_m(\mathbf{e}, \mathbf{f}_s)}_{=b(\mathbf{e},\mathbf{f}_s)} \Bigg\}$$

$$= \arg\max_{\mathbf{e} \in \mathbf{C}_s} \big\{ \underbrace{a(\mathbf{e}, \mathbf{f}_s) + \gamma \cdot b(\mathbf{e}, \mathbf{f}_s)}_{(*)} \big\} \quad (5)$$

Hence, the total score $(*)$ for any candidate translation corresponds to a line in the plane with γ as the independent variable. For any particular choice of γ, the decoder seeks that translation which yields the largest score and therefore corresponds to the topmost line segment.

Overall, the candidate repository $\mathbf{C}_s$ defines K lines where each line may be divided into at most K line segments due to possible intersections with the other $K - 1$ lines. The sequence of the topmost line segments constitute the *upper envelope* which is the pointwise maximum over all lines induced by $\mathbf{C}_s$. The upper envelope is a convex hull and can be inscribed with a convex polygon whose edges are the segments of a piecewise linear function in γ (Papineni, 1999; Och, 2003):

$$\mathsf{Env}(\mathbf{f}) = \max_{\mathbf{e} \in \mathbf{C}} \big\{ a(\mathbf{e}, \mathbf{f}) + \gamma \cdot b(\mathbf{e}, \mathbf{f}) : \gamma \in \mathbb{R} \big\} \quad (6)$$

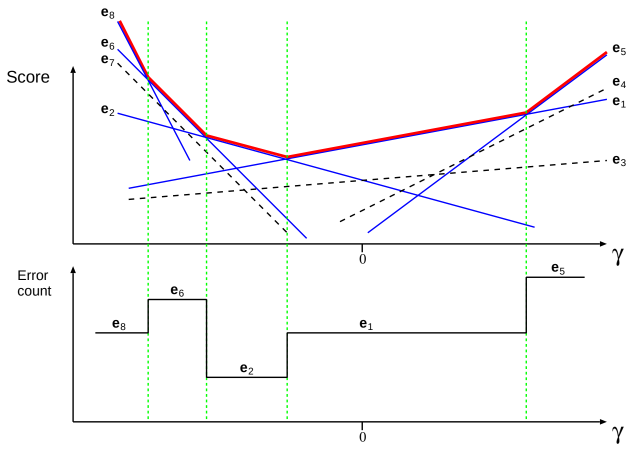


Figure 1: *The upper envelope (bold, red curve) for a set of lines is the convex hull which consists of the topmost line segments. Each line corresponds to a candidate translation and is thus related to a certain error count. Envelopes can efficiently be computed with Algorithm 1.*

The importance of the upper envelope is that it provides a compact encoding of all possible outcomes that a rescoring of $\mathbf{C}_s$ may yield if the parameter set λ_1^M is moved along the chosen direction. Once the upper envelope has been determined, we can project its constituent line segments onto the error counts of the corresponding candidate translations (cf. Figure 1). This projection is independent of *how* the envelope is generated and can therefore be applied to any set of line segments[1].

An effective means to compute the upper envelope is a *sweep line* algorithm which is often used in computational geometry to determine the intersection points of a sequence of lines or line segments (Bentley and Ottmann, 1979). The idea is to shift ("sweep") a vertical ray from $-\infty$ to $+\infty$ over the plane while keeping track of those points where two or more lines intersect. Since the upper envelope is fully specified by the topmost line segments, it suffices to store the following components for each line object ℓ: the x-intercept $\ell.x$ with the left-adjacent line, the slope $\ell.m$, and the y-intercept $\ell.y$; a fourth component, $\ell.t$, is used to store the candidate translation. Algorithm 1 shows the pseudo code for a sweep line algorithm which reduces an input array `a[0..K-1]` consisting of the K line objects of the candidate repository $\mathbf{C}_s$ to its upper envelope. By construction, the upper envelope consists of at most K line segments. The endpoints of each line

Algorithm 1 SweepLine

input: array `a[0..K-1]` containing lines
output: upper envelope of `a`

```
sort(a:$m$);
j = 0; K = size(a);
for (i = 0; i < K; ++i) {
  $\ell$ = a[i];
  $\ell.x$ = -$\infty$;
  if (0 < j) {
    if (a[j-1].m == $\ell.m$) {
      if ($\ell.y$ <= a[j-1].y) continue;
      --j;
    }
    while (0 < j) {
      $\ell.x$ = ($\ell.y$ - a[j-1].y)/
            (a[j-1].m - $\ell.m$);
      if (a[j-1].x < $\ell.x$) break;
      --j;
    }
    if (0 == j) $\ell.x$ = -$\infty$;
    a[j++] = $\ell$;
  } else a[j++] = $\ell$;
}
a.resize(j);
return a;
```

segment define the interval boundaries at which the decision made by the decoder will change. Hence, as γ increases from $-\infty$ to $+\infty$, we will see that the most probable translation hypothesis will change whenever γ passes an intersection point.

Let $\gamma_1^{\mathbf{f}_s} < \gamma_2^{\mathbf{f}_s} < \ldots < \gamma_{N_s}^{\mathbf{f}_s}$ denote the sequence of interval boundaries and let $\Delta E_1^{\mathbf{f}_s}, \Delta E_2^{\mathbf{f}_s}, \ldots, \Delta E_{N_s}^{\mathbf{f}_s}$ denote the corresponding sequence of changes in the error count where $\Delta E_n^{\mathbf{f}_s}$ is the amount by which the error count will change if γ is moved from a point in $[\gamma_{n-1}^{\mathbf{f}_s}, \gamma_n^{\mathbf{f}_s})$ to a point in $[\gamma_n^{\mathbf{f}_s}, \gamma_{n+1}^{\mathbf{f}_s})$. Both sequences together provide an exhaustive representation of the unsmoothed error surface for the sentence $\mathbf{f}_s$ along the line $\lambda_1^M + \gamma \cdot d_1^M$. The error surface for the whole training corpus is obtained by merging the interval boundaries (and their corresponding error counts) over all sentences in the training corpus. The optimal γ can then be found by traversing the merged error surface and choosing a point from the interval where the total error reaches its minimum.

After the parameter update, $\hat{\lambda}_1^M = \lambda_1^M + \gamma_{\text{opt}} \cdot d_1^M$, the decoder may find new translation hypotheses which are merged into the candidate repositories if they are ranked among the top N candidates. The relation $K = N$ holds therefore only in the first iteration. From the second iteration on, K is usually larger than N. The sequence of line optimizations and decodings is repeated until (1) the candidate repositories remain unchanged and (2) $\gamma_{\text{opt}} = 0$.

[1] For lattice MERT, it will therefore suffice to find an efficient way to compute the upper envelope over all translations that are encoded in a phrase graph.

3 Minimum Error Rate Training on Lattices

In this section, the algorithm for computing the upper envelope on N-best lists is extended to phrase lattices. For a description on how to generate lattices, see (Ueffing et al., 2002).

Formally, a phrase lattice for a source sentence $\mathbf{f}$ is defined as a connected, directed acyclic graph $\mathcal{G}_\mathbf{f} = (\mathcal{V}_\mathbf{f}, \mathcal{E}_\mathbf{f})$ with vertice set $\mathcal{V}_\mathbf{f}$, unique source and sink nodes $s, t \in \mathcal{V}_\mathbf{f}$, and a set of arcs $\mathcal{E}_\mathbf{f} \subset \mathcal{V}_\mathbf{f} \times \mathcal{V}_\mathbf{f}$. Each arc is labeled with a phrase $\varphi_{ij} = e_{i_1}, ..., e_{i_j}$ and the (local) feature function values $h_1^M(\varphi_{ij}, \mathbf{f})$. A path $\pi = (v_0, \varepsilon_0, v_1, \varepsilon_1, ..., \varepsilon_{n-1}, v_n)$ in $\mathcal{G}_\mathbf{f}$ (with $\varepsilon_i \in \mathcal{E}_\mathbf{f}$ and $v_i, v_{i+1} \in \mathcal{V}_\mathbf{f}$ as the tail and head of ε_i, $0 \leqslant i < n$) defines a partial translation $\mathbf{e}_\pi$ of $\mathbf{f}$ which is the concatenation of all phrases along this path. The corresponding feature function values are obtained by summing over the arc-specific feature function values:

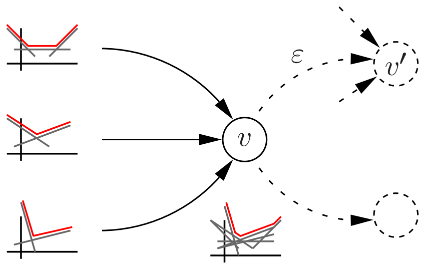
$$\pi : \underset{v_0}{\bullet} \xrightarrow[h_1^M(\varphi_{0,1}, \mathbf{f})]{\varphi_{0,1}} \underset{v_1}{\bullet} \xrightarrow[h_1^M(\varphi_{1,2}, \mathbf{f})]{\varphi_{1,2}} \cdots \xrightarrow[h_1^M(\varphi_{n-1,n}, \mathbf{f})]{\varphi_{n-1,n}} \underset{v_n}{\bullet}$$

$$\mathbf{e}_\pi = \bigcirc_{\substack{i,j:\\ v_i \rightarrow v_j \in \pi}} \varphi_{ij} = \varphi_{0,1} \circ ... \circ \varphi_{n-1,n}$$

$$h_1^M(\mathbf{e}_\pi, \mathbf{f}) = \sum_{\substack{i,j:\\ v_i \rightarrow v_j \in \pi}} h_1^M(\varphi_{ij}, \mathbf{f})$$

In the following, we use the notation $\mathsf{in}(v)$ and $\mathsf{out}(v)$ to refer to the set of incoming and outgoing arcs for a node $v \in \mathcal{V}_\mathbf{f}$. Similarly, $\mathsf{head}(\varepsilon)$ and $\mathsf{tail}(\varepsilon)$ denote the head and tail of $\varepsilon \in \mathcal{E}_\mathbf{f}$.

To develop the algorithm for computing the upper envelope of *all* translation hypotheses that are encoded in a phrase lattice, we first consider a node $v \in \mathcal{V}_\mathbf{f}$ with some incoming and outgoing arcs:



Each path that starts at the source node s and ends in v defines a partial translation hypothesis which can be represented as a line (cf. Eq. (5)). We now assume that the upper envelope for these partial translation hypotheses is known. The lines that constitute this envelope shall be denoted by $f_1, ..., f_N$. Next we consider continuations of these partial translation candidates by following one of the outgoing arcs $\varepsilon \in \mathsf{out}(v)$. Each such arc defines another line denoted by $g(\varepsilon)$. If we add the slope and y-intercept of $g(\varepsilon)$ to each line in the set $\{f_1, ..., f_N\}$, then the upper envelope will be constituted by segments of $f_1 + g(\varepsilon), ..., f_N + g(\varepsilon)$. This operation neither changes the number of line segments nor their relative order in the envelope, and therefore it preserves the structure of the convex hull. As a consequence, we can propagate the resulting envelope over an outgoing arc ε to a successor node $v' = \mathsf{head}(\varepsilon)$. Other incoming arcs for v' may be associated with different upper envelopes, and all that remains is to merge these envelopes into a single combined envelope. This is, however, easy to accomplish since the combined envelope is simply the convex hull of the union over the line sets which constitute the individual envelopes. Thus, by merging the arrays that store the line segments for the incoming arcs and applying Algorithm 1 to the resulting array we obtain the combined upper envelope for *all* partial translation candidates that are associated with paths starting at the source node s and ending in v'. The correctness of this procedure is based on the following two observations:

(1) A single translation hypothesis cannot constitute multiple line segments of the same envelope. This is because translations associated with different line segments are path-disjoint.

(2) Once a partial translation has been discarded from an envelope because its associated line $\tilde{f}$ is completely covered by the topmost line segments of the convex hull, there is no path continuation that could bring back $\tilde{f}$ into the upper envelope

Algorithm 2 Lattice Envelope

input: a phrase lattice $\mathcal{G}_\mathbf{f} = (\mathcal{V}_\mathbf{f}, \mathcal{E}_\mathbf{f})$
output: upper envelope of $\mathcal{G}_\mathbf{f}$

```
a = $\emptyset$;
L = $\emptyset$;
TopSort($\mathcal{G}_\mathbf{f}$);
for $v$ = $s$ to $t$ do {
  a = SweepLine( $\bigcup_{\varepsilon \in \mathsf{in}(v)}$ L[$\varepsilon$]);
  foreach ($\varepsilon \in \mathsf{in}(v)$)
    L.delete($\varepsilon$);
  foreach ($\varepsilon \in \mathsf{out}(v)$) {
    L[$\varepsilon$] = a;
    for (i = 0; i < a.size(); ++i) {
      L[$\varepsilon$][i].m = a[i].m + $\sum_m d_m h_m(\varepsilon, \mathbf{f})$;
      L[$\varepsilon$][i].y = a[i].y + $\sum_m \lambda_m h_m(\varepsilon, \mathbf{f})$;
      L[$\varepsilon$][i].p = a[i].p $\circ \varphi_{v,\mathsf{head}(\varepsilon)}$;
    }
  }
}
return a;
```

again. Proof: Suppose that such a continuation exists, then this continuation can be represented as a line g, and since $\tilde{f}$ has been discarded from the envelope, the path associated with g must also be a valid continuation for the line segments $f_1, ..., f_N$ that constitute the envelope. Thus it follows that $\max(f_1 + g, ..., f_N + g) = \max(f_1, ..., f_N) + g < \tilde{f} + g$ for some $\gamma \in \mathbb{R}$. This, however, is in contradiction with the premise that $\tilde{f} < \max(f_1, ..., f_N)$ for all $\gamma \in \mathbb{R}$.

To keep track of the phrase expansions when propagating an envelope over an outgoing arc $\varepsilon \in \mathsf{tail}(v)$, the phrase label $\varphi_{v,\mathsf{head}(\varepsilon)}$ has to be appended from the right to all partial translation hypotheses in the envelope. The complete algorithm then works as follows: First, all nodes in the phrase lattice are sorted in topological order. Starting with the source node, we combine for each node v the upper envelopes that are associated with v's incoming arcs by merging their respective line arrays and reducing the merged array into a combined upper envelope using Algorithm 1. The combined envelope is then propagated over the outgoing arcs by associating each $\varepsilon \in \mathsf{out}(v)$ with a copy of the combined envelope. This copy is modified by adding the parameters (slope and y-intercept) of the line $g(\varepsilon)$ to the envelope's constituent line segments. The envelopes of the incoming arcs are no longer needed and can be deleted in order to release memory. The envelope computed at the sink node is by construction the convex hull over all translation hypotheses represented in the lattice, and it compactly encodes those candidates which maximize the decision rule Eq. (1) for any point along the line $\lambda_1^M + \gamma \cdot d_1^M$. Algorithm 2 shows the pseudo code. Note that the component $\ell.x$ does not change and therefore requires no update.

It remains to verify that the suggested algorithm is efficient in both running time and memory. For this purpose, we first analyze the complexity of Algorithm 1 and derive from it the running time of Algorithm 2.

After sorting, each line object in Algorithm 1 is visited at most three times. The first time is when it is picked by the outer loop. The second time is when it either gets discarded or when it terminates the inner loop. Whenever a line object is visited for the third time, it is irrevocably removed from the envelope. The runtime complexity is therefore dominated by the initial sorting and amounts to $\mathcal{O}(K \log K)$

Topological sort on a phrase lattice $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ can be performed in time $\Theta(|\mathcal{V}| + |\mathcal{E}|)$. As will be shown in Section 4, the size of the upper envelope for $\mathcal{G}$ can never exceed the size of the arc set $\mathcal{E}$. The same holds for any subgraph $\mathcal{G}_{[s,v]}$ of $\mathcal{G}$ which is induced by the paths that connect the source node s with $v \in \mathcal{V}$. Since the envelopes propagated from the source to the sink node can only increase linearly in the number of previously processed arcs, the total running time amounts to a worst case complexity of $\mathcal{O}(|\mathcal{V}| \cdot |\mathcal{E}| \log |\mathcal{E}|)$.

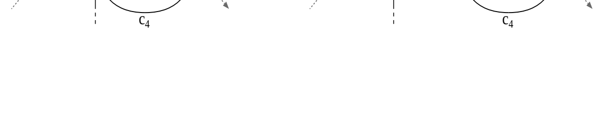
4 Upper Bound for Size of Envelopes

The memory efficiency of the suggested algorithm results from the following theorem which provides a novel upper bound for the number of cost minimizing paths in a directed acyclic graph with arc-specific affine cost functions. The bound is not only meaningful for proving the space efficiency of lattice MERT, but it also provides deeper insight into the structure and complexity of the unsmoothed error surface induced by log-linear models. Since we are examining a special class of shortest paths problems, we will invert the sign of each local feature function value in order to turn the feature scores into corresponding costs. Hence, the objective of finding the best translation hypotheses in a phrase lattice becomes the problem of finding all cost-minimizing paths in a graph with affine cost functions.

Theorem: *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a connected directed acyclic graph with vertex set $\mathcal{V}$, unique source and sink nodes $s, t \in \mathcal{V}$, and an arc set $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ in which each arc $\varepsilon \in \mathcal{E}$ is associated with an affine cost function $c_\varepsilon(\gamma) = a_\varepsilon \cdot \gamma + b_\varepsilon$, $a_\varepsilon, b_\varepsilon \in \mathbb{R}$. Counting ties only once, the cardinality of the union over the sets of all cost-minimizing paths for all $\gamma \in \mathbb{R}$ is then upper-bounded by $|\mathcal{E}|$:*

$$\left| \bigcup_{\gamma \in \mathbb{R}} \{\pi \ : \ \pi = \pi(\mathcal{G}; \gamma) \text{ is a cost-minimizing path in } \mathcal{G} \text{ given } \gamma\} \right| \leqslant |\mathcal{E}| \quad (7)$$

Proof: The proposition holds for the empty graph as well as for the case that $\mathcal{V} = \{s, t\}$ with all arcs $\varepsilon \in \mathcal{E}$ joining the source and sink node. Let $\mathcal{G}$ therefore be a larger graph. Then we perform an s-t cut and split $\mathcal{G}$ into two subgraphs $\mathcal{G}_1$ (left subgraph) and $\mathcal{G}_2$ (right subgraph). Arcs spanning the section boundary are duplicated (with the costs of the copied arcs in $\mathcal{G}_2$ being set to zero) and connected with a newly added head or tail node:



The zero-cost arcs in $\mathcal{G}_2$ that emerged from the duplication process are contracted, which can be done without loss of generality because zero-cost arcs do not affect the total costs of paths in the lattice. The contraction essentially amounts to a removal of arcs and is required in order to ensure that the sum of edges in both subgraphs does not exceed the number of edges in $\mathcal{G}$. All nodes in $\mathcal{G}_1$ with out-degree zero are then combined into a single sink node t_1. Similarly, nodes in $\mathcal{G}_2$ whose in-degree is zero are combined into a single source node s_2. Let N_1 and N_2 denote the number of arcs in $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively. By construction, $N_1 + N_2 = |\mathcal{E}|$. Both subgraphs are smaller than $\mathcal{G}$ and thus, due to the induction hypothesis, their lower envelopes consist of at most N_1 and N_2 line segments, respectively. We further notice that either envelope is a convex hull whose constituent line segments inscribe a convex polygon, in the following denoted by $\mathcal{P}_1$ and $\mathcal{P}_2$. Now, we combine both subgraphs into a single graph $\mathcal{G}'$ by merging the sink node t_1 in $\mathcal{G}_1$ with the source node s_2 in $\mathcal{G}_2$. The merged node is an *articulation point* whose removal would disconnect both subgraphs, and hence, all paths in $\mathcal{G}'$ that start at the source node s and stop in the sink node t lead through this articulation point. The graph $\mathcal{G}'$ has at least as many cost minimizing paths as $\mathcal{G}$, although these paths as well as their associated costs might be different from those in $\mathcal{G}$. The additivity of the cost function and the articulation point allow us to split the costs for any path from s to t into two portions: the first portion can be attributed to $\mathcal{G}_1$ and must be a line inside $\mathcal{P}_1$; the remainder can be attributed to $\mathcal{G}_2$ and must therefore be a line inside $\mathcal{P}_2$. Hence, the total costs for any path in $\mathcal{G}'$ can be bounded by the convex hull of the superposition of $\mathcal{P}_1$ and $\mathcal{P}_2$. This convex hull is again a convex polygon which consists of at most $N_1 + N_2$ edges, and therefore, the number of cost minimizing paths in $\mathcal{G}'$ (and thus also in $\mathcal{G}$) is upper bounded by $N_1 + N_2$. $\square$

Corollary: *The upper envelope for a phrase lattice $\mathcal{G}_{\mathbf{f}} = (\mathcal{V}_{\mathbf{f}}, \mathcal{E}_{\mathbf{f}})$ consists of at most $|\mathcal{E}_{\mathbf{f}}|$ line segments.* This bound can even be refined and one obtains (proof omitted) $|\mathcal{E}| - |\mathcal{V}| + 2$. Both bounds are tight.

This result may seem somewhat surprising as it states that, independent of the choice of the direction along which the line optimization is performed, the structure of the error surface is far less complex than one might expect based on the huge number of alternative translation candidates that are represented in the lattice and thus contribute to the error surface. In fact, this result is a consequence of using a log-linear model which constrains how costs (or scores, respectively) can evolve due to hypothesis expansion. If instead quadratic cost functions were used, the size of the envelopes could not be limited in the same way. The above theorem does not, however, provide any additional guidance that would help to choose more promising directions in the line optimization algorithm to find better local optima. To alleviate this problem, the following section lists some best practices that we found to be useful in the context of MERT.

5 Practical Aspects

This section addresses some techniques that we found to be beneficial in order to improve the performance of MERT.

(1) **Random Starting Points:** To prevent the line optimization algorithm from stopping in a poor local optimum, MERT explores additional starting points that are randomly chosen by sampling the parameter space.

(2) **Constrained Optimization:** This technique allows for limiting the range of some or all feature function weights by defining *weights restrictions*. The weight restriction for a feature function h_m is specified as an interval $\mathcal{R}_m = [l_m, r_m]$, $l_m, r_m \in \mathbb{R} \cup \{-\infty, +\infty\}$ which defines the admissible region from which the feature function weight λ_m can be chosen. If the line optimization is performed under the presence of weights restrictions, γ needs to be chosen such that the following constraint holds:

$$l_1^M \leqslant \lambda_1^M + \gamma \cdot d_1^M \leqslant r_1^M \tag{8}$$

(3) **Weight Priors:** Weight priors give a small (positive or negative) boost ω on the objective function if the new weight is chosen such that it matches a certain target value λ_m^*:

$$\gamma_{\text{opt}} = \arg\min_{\gamma} \Big\{ \sum_s E\big(\mathbf{r}_s, \hat{e}(\mathbf{f}_s; \gamma)\big) + \sum_m \delta(\lambda_m + \gamma \cdot d_m, \lambda_m^*) \cdot \omega \Big\} \tag{9}$$

A *zero-weights prior* ($\lambda_m^* = 0$) provides a means of doing feature selection since the weight of a feature function which is not discriminative will be set to zero. An *initial-weights prior* ($\lambda_m^* = \lambda_m$) can be used to confine changes in the parameter update with the consequence that the new parameter may be closer to the initial weights set. Initial weights priors are useful in cases where the starting weights already yield a decent baseline.

(4) **Interval Merging:** The interval $[\gamma_i^{\mathbf{f}_s}, \gamma_{i+1}^{\mathbf{f}_s})$ of a translation hypothesis can be merged with the interval $[\gamma_{i-1}^{\mathbf{f}_s}, \gamma_i^{\mathbf{f}_s})$ of its left-adjacent translation hypothesis if the corresponding change in the error count $\Delta E_i^{\mathbf{f}_s} = 0$. The resulting interval $[\gamma_{i-1}^{\mathbf{f}_s}, \gamma_{i+1}^{\mathbf{f}_s})$ has a larger range, and the choice of γ_{opt} may be more reliable.

(5) **Random Directions:** If the directions chosen in the line optimization algorithm are the coordinate axes of the M-dimensional parameter space, each iteration will result in the update of a single feature function only. While this update scheme provides a ranking of the feature functions according to their discriminative power (each iteration picks the feature function for which changing the corresponding weight yields the highest gain), it does not take possible correlations between the feature functions into account. As a consequence, the optimization procedure may stop in a poor local optimum. On the other hand, it is difficult to compute a direction that decorrelates two or more correlated feature functions. This problem can be alleviated by exploring a large number of random directions which update many feature weights simultaneously. The random directions are chosen as the lines which connect some randomly distributed points on the surface of an M-dimensional hypersphere with the hypersphere's center. The center of the hypersphere is defined as the initial parameter set.

6 Related Work

As suggested in (Och, 2003), an alternative method for the optimization of the unsmoothed error count is Powell's algorithm combined with a grid-based line optimization (Press et al., 2007, p. 509). In (Zens et al., 2007), the MERT criterion is optimized on N-best lists using the Downhill Simplex algorithm (Press et al., 2007, p. 503). The optimization procedure allows for optimizing other objective function as, e.g., the expected BLEU score. A weakness of the Downhill Simplex algorithm is, however, its decreasing robustness for optimization problems in more than 10 dimensions. A different approach to minimize the expected BLEU score is suggested in (Smith and Eisner, 2006) who use deterministic annealing to gradually turn the objective function from a convex entropy surface into the more complex risk surface. A large variety of different search strategies for MERT are investigated in (Cer et al., 2008), which provides many fruitful insights into the optimization process. In (Duh and Kirchhoff, 2008), MERT is used to boost the BLEU score on N-best re-ranking tasks. The incorporation of a large number of sparse feature functions is described in (Watanabe et al., 2007). The paper investigates a perceptron-like online large-margin training for statistical machine translation. The described approach is reported to yield significant improvements on top of a baseline system which employs a small number of feature functions whose weights are optimized under the MERT criterion. A study which is complementary to the upper bound on the size of envelopes derived in Section 4 is provided in (Elizalde and Woods, 2006) which shows that the number of inference functions of any graphical model as, for instance, Bayesian networks and Markov random fields is polynomial in the size of the model if the number of parameters is fixed.

Table 1: *Corpus statistics for three text translation sets: Arabic-to-English (aren), Chinese-to-English (zhen), and English-to-Chinese (enzh). Development and test data are compiled from evaluation data used in past NIST Machine Translation Evaluations.*

data set	collection	# of sentences		
		aren	zhen	enzh
dev1	nist02	1043	878	–
dev2	nist04	1353	1788	–
blind	nist08	1360	1357	1859

7 Experiments

Experiments were conducted on the NIST 2008 translation tasks under the conditions of the constrained data track for the language pairs Arabic-to-English (aren), English-to-Chinese (enzh), and Chinese-to-English (zhen). The development corpora were compiled from test data used in the 2002 and 2004 NIST evaluations. Each corpus set provides 4 reference translations per source sentence. Table 1 summarizes some corpus statistics.

Table 2: *BLEU score results on the NIST-08 test set obtained after 25 iterations using N-best MERT or 5 iterations using lattice MERT, respectively.*

		dev1+dev2		blind	
task	loss	N-best	lattice	N-best	lattice
aren	MBR	56.6	57.4	42.9	43.9
	0-1	56.7	57.4	42.8	43.7
enzh	MBR	39.7	39.6	36.5	38.8
	0-1	40.4	40.5	35.1	37.6
zhen	MBR	39.5	39.7	27.5	28.2
	0-1	39.6	39.6	27.0	27.6

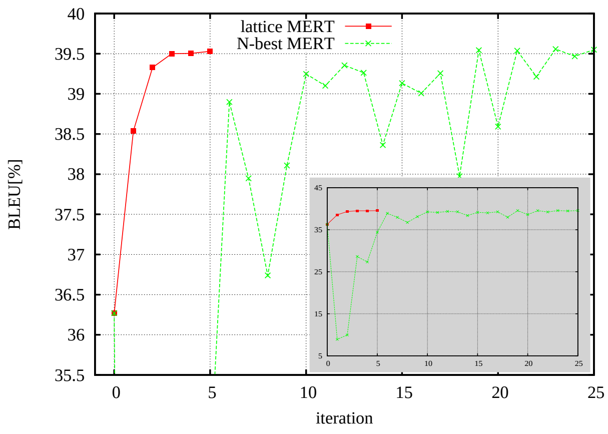


Figure 2: *BLEU scores for N-best MERT and lattice MERT after each decoding step on the zhen-dev1 corpus. The grey shaded subfigure shows the complete graph including the bottom part for N-best MERT.*

Translation results were evaluated using the mixed-case BLEU score metric in the implementation as suggested by (Papineni et al., 2001).

Translation results were produced with a state-of-the-art phrase-based SMT system which uses EM-trained word alignment models (IBM1, HMM) and a 5-gram language model built from the Web-1T collection[2]. Translation hypotheses produced on the blind test data were reranked using the *Minimum-Bayes Risk* (MBR) decision rule (Kumar and Byrne, 2004; Tromble et al., 2008). Each system uses a log-linear combination of 20 to 30 feature functions.

In a first experiment, we investigated the convergence speed of lattice MERT and N-best MERT.

[2]http://www.ldc.upenn.edu, catalog entry: LDC2006T13

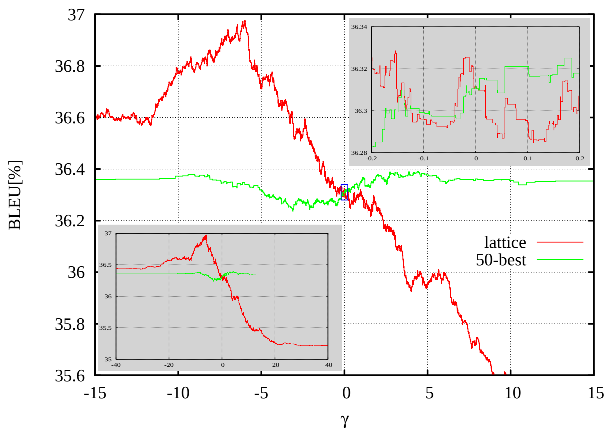


Figure 3: *Error surface of the phrase penalty feature after the first iteration on the zhen-dev1 corpus.*

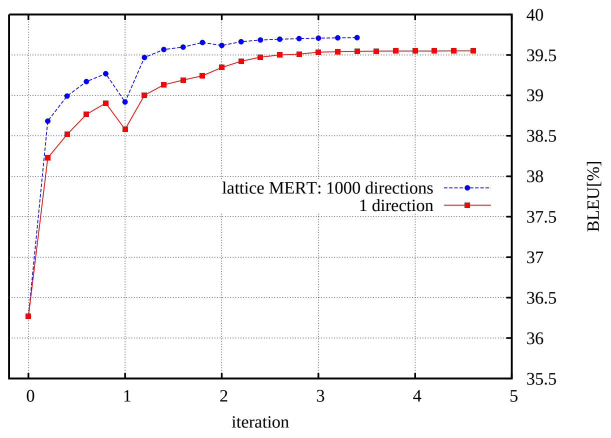


Figure 4: *BLEU scores on the zhen-dev1 corpus for lattice MERT with additional directions.*

Figure 2 shows the evolution of the BLEU score in the course of the iteration index on the zhen-dev1 corpus for either method. In each iteration, the training procedure translates the development corpus using the most recent weights set and merges the top ranked candidate translations (either represented as phrase lattices or N-best lists) into the candidate repositories before the line optimization is performed. For N-best MERT, we used $N = 50$ which yielded the best results. In contrast to lattice MERT, N-best MERT optimizes *all* dimensions in each iteration and, in addition, it also explores a large number of random starting points before it re-decodes and expands the hypothesis set. As is typical for N-best MERT, the first iteration causes a dramatic performance loss caused by overadapting the candidate repositories, which amounts to more than 27.3 BLEU points. Although this performance loss is recouped after the 5th iteration, the initial decline makes the line optimization under N-best MERT more fragile since the optimum found at the end of the training procedure is affected by the initial performance drop rather than by the choice of the initial start weights. Lattice MERT on the other hand results in a significantly faster convergence speed and reaches its optimum already in the 5th iteration. For lattice MERT, we used a graph density of 40 arcs per phrase which corresponds to an N-best size of more than two octillion $(2 \cdot 10^{27})$ entries. This huge number of alternative candidate translations makes updating the weights under lattice MERT more reliable and robust and, compared to N-best MERT, it becomes less likely that the same feature weight needs to be picked again and adjusted in subsequent iterations. Figure 4 shows the evolution of the BLEU score on the zhen-dev1 corpus using

Table 3: *BLEU score results on the NIST-08 tests set obtained after 5 iterations using lattice MERT with different numbers of random directions in addition to the optimization along the coordinate axes.*

task	# random directions	dev1+dev2 0-1	dev1+dev2 MBR	blind 0-1	blind MBR
aren	–	57.4	57.4	43.7	43.9
	1000	57.6	57.7	43.9	44.5
zhen	–	39.6	39.7	27.6	28.2
	500	39.5	39.9	27.9	28.3

lattice MERT with 5 weights updates per iteration. The performance drop in iteration 1 is also attributed to overfitting the candidate repository. The decline of less than 0.5% in terms of BLEU is, however, almost negligible compared to the performance drop of more than 27% in case of N-best MERT. The vast number of alternative translation hypotheses represented in a lattice also increases the number of phase transitions in the error surface, and thus prevents MERT from selecting a low performing feature weights set at early stages in the optimization procedure. This is illustrated in Figure 3, where lattice MERT and N-best MERT find different optima for the weight of the phrase penalty feature function after the first iteration. Table 2 shows the BLEU score results on the NIST 2008 blind test using the combined dev1+dev2 corpus as training data. While only the aren task shows improvements on the development data, lattice MERT provides consistent gains over N-best MERT on all three blind test sets. The reduced performance for N-best MERT is a consequence of the performance drop in the first iteration which causes the final weights to be far off from the initial parameter set. This can impair the ability of N-best MERT to generalize to unseen data if the initial weights are already capable of producing a decent baseline. Lattice MERT on the other hand can produce weights sets which are closer to the initial weights and thus more likely to retain the ability to generalize to unseen data. It could therefore be worthwhile to investigate whether a more elaborated version of an initial-weights prior allows for alleviating this effect in case of N-best MERT. Table 3 shows the effect of optimizing the feature function weights along some randomly chosen directions in addition to the coordinate axes. The different local optima found on the development set by using random directions result in additional gains on the blind test sets and range from 0.1% to 0.6% absolute in terms of BLEU.

8 Summary

We presented a novel algorithm that allows for efficiently constructing and representing the unsmoothed error surface over all sentence hypotheses that are represented in a phrase lattice. The proposed algorithm was used to train the feature function weights of a log-linear model for a statistical machine translation system under the *Minimum Error Rate Training* (MERT) criterion. Lattice MERT was shown analytically and experimentally to be superior over N-best MERT, resulting in significantly faster convergence speed and a reduced number of decoding steps. While the approach was used to optimize the model parameters of a single machine translation system, there are many other applications in which this framework can be useful, too. One possible usecase is the computation of consensus translations from the outputs of multiple machine translation systems where this framework allows us to estimate the system prior weights directly on confusion networks (Rosti et al., 2007; Macherey and Och, 2007). It is also straightforward to extend the suggested method to hypergraphs and forests as they are used, e.g., in hierarchical and syntax-augmented systems (Chiang, 2005; Zollmann and Venugopal, 2006). Our future work will therefore focus on how much system combination and syntax-augmented machine translation can benefit from lattice MERT and to what extent feature function weights can robustly be estimated using the suggested method.

References

J. L. Bentley and T. A. Ottmann. 1979. Algorithms for reporting and counting geometric intersections. *IEEE Trans. on Computers*, C-28(9):643–647.

D. Cer, D. Jurafsky, and C. D. Manning. 2008. Regularization and Search for Minimum Error Rate Training. In *Proceedings of the Third Workshop on Statistical Machine Translation, 46th Annual Meeting of the Association of Computational Linguistics: Human Language Technologies (ACL-2008 HLT)*, pages 26–34, Columbus, OH, USA, June.

D. Chiang. 2005. A Hierarchical Phrase-based Model for Statistical Machine Translation. In *ACL-2005*, pages 263–270, Ann Arbor, MI, USA, June.

K. Duh and K. Kirchhoff. 2008. Beyond Log-Linear Models: Boosted Minimum Error Rate Training for N-best Re-ranking. In *Proceedings of the Third Workshop on Statistical Machine Translation, 46th Annual Meeting of the Association of Computational Linguistics: Human Language Technologies (ACL-2008 HLT)*, pages 37–40, Columbus, OH, USA, June.

S. Elizalde and K. Woods. 2006. Bounds on the Number of Inference Functions of a Graphical Model, October. `arXiv:math/0610233v1`.

S. Kumar and W. Byrne. 2004. Minimum Bayes-Risk Decoding for Statistical Machine Translation. In *Proc. HLT-NAACL*, pages 196–176, Boston, MA, USA, May.

W. Macherey and F. J. Och. 2007. An Empirical Study on Computing Consensus Translations from Multiple Machine Translation Systems. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 986–995, Prague, Czech Republic, June.

F. J. Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 160–167, Sapporo, Japan, July.

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

K. A. Papineni. 1999. Discriminative training via linear programming. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 2, pages 561–564, Phoenix, AZ, March.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 2007. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge, UK, third edition.

A. V. Rosti, N. F. Ayan, B. Xiang, S. Matsoukas, R. Schwartz, and B. Dorr. 2007. Combining outputs from multiple machine translation systems. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 228–235, Rochester, New York, April. Association for Computational Linguistics.

D. A. Smith and J. Eisner. 2006. Minimum Risk Annealing for Training Log-linear Models. In *21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (Coling/ACL-2006)*, pages 787–794, Sydney, Australia, July.

R. Tromble, S. Kumar, F. J. Och, and W. Macherey. 2008. Lattice minimum bayes-risk decoding for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 10, Waikiki, Honolulu, Hawaii, USA, October.

N. Ueffing, F. J. Och, and H. Ney. 2002. Generation of word graphs in statistical machine translation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 156–163, Philadelphia, PE, July.

T. Watanabe, J. Suzuki, H. Tsukada, and H. Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 764–773, Prague, Czech Republic.

R. Zens, S. Hasan, and H. Ney. 2007. A Systematic Comparison of Training Criteria for Statistical Machine Translation. In *Proceedings of the 2007 Conference on Empirical Methods in Natural Language Processing*, Prague, Czech Republic, June. Association for Computational Linguistics.

A. Zollmann and A. Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *NAACL '06: Proceedings of the 2006 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 138–141, New York, NY, June. Association for Computational Linguistics.

Syntactic Models for Structural Word Insertion and Deletion

Arul Menezes and **Chris Quirk**
Microsoft Research
One Microsoft Way, Redmond, WA 98052, USA
{arulm, chrisq}@microsoft.com

Abstract

An important problem in translation neglected by most recent statistical machine translation systems is insertion and deletion of words, such as function words, motivated by linguistic structure rather than adjacent lexical context. Phrasal and hierarchical systems can only insert or delete words in the context of a larger phrase or rule. While this may suffice when translating in-domain, it performs poorly when trying to translate broad domains such as web text. Various syntactic approaches have been proposed that begin to address this problem by learning lexicalized and unlexicalized rules. Among these, the treelet approach uses unlexicalized order templates to model ordering separately from lexical choice. We introduce an extension to the latter that allows for structural word insertion and deletion, without requiring a lexical anchor, and show that it produces gains of more than 1.0% BLEU over both phrasal and baseline treelet systems on broad domain text.

1 Introduction

Among the phenomena that are modeled poorly by modern SMT systems is the insertion and deletion of words, such as function words, that are motivated by the divergent linguistic structure between source and target language. To take the simplest of examples, the English noun compound "*file name*" would typically be translated into Spanish as "*nombre de archivo*", which requires the insertion of the preposition "*de*". Conversely, when translating from Spanish to English, the "*de*" must be deleted. At first glance, the problem may seem trivial, yet the presence and position of these function words can have crucial impact on the adequacy and fluency of translation.

In particular, function words are often used to denote key semantic information. They may be used to denote case information, in languages such as Japanese. Failing to insert the proper case marker may render a sentence unreadable or significantly change its meaning. Learning these operations can be tricky for MT models best suited to contiguous word sequences. From a fluency standpoint, proper insertion of determiners and prepositions can often make the difference between laughably awkward output and natural sounding translations; consider the output "*it's a cake piece*" as opposed to "*it's a piece of cake*".

Furthermore, since missing or spurious function words can confuse the target language model, handling these words properly can have an impact beyond the words themselves.

This paper focuses on methods of inserting and deleting words based on syntactic cues, to be used in the context of a syntax-informed translation system. While the models we build are relatively simple and the underlying templates are easy to extract, they add significant generalization ability to the base translation system, and result in significant gains.

2 Background

As a motivating example, let us return to the English/Spanish pair "*file name*" and "*nombre de archivo*". In principle, we would want a machine translation system to be capable of learning the following general transformation:

$$\text{“ NOUN}_1\text{ NOUN}_2\text{”} \rightarrow \text{“ NOUN}_2\text{ } \textit{de} \text{ NOUN}_1\text{”} \qquad (1)$$

Yet even this simple example is beyond the capabilities of many common approaches.

The heavily lexicalized approaches of phrasal systems (Koehn et al., 2003), are inherently incapable of this generalization. As a proxy, they acquire phrase pairs such as "*nombre de archivo*" → "*file name*", "*nombre de*" → "*name*" and "*de archivo*" → "*file*". Note that the inserted word is attached to adjacent context word(s). When the test set vocabulary has significant overlap with the training vocabulary, the correct translation can often be assembled based on the head or the modifying noun. However, as we show in this paper, this is woefully inadequate when translating truly out-of-domain input.

In principle, phrase-based translation systems may employ insertion phrase pairs such as

"[NULL]" → "*de*" (2)

but the ungrounded nature of this transformation makes its use during decoding difficult. Since there are no constraints on where such a rule may apply and the rule does not consume any input words, the decoder must attempt these rules at every point in the search.

The reverse operation

"*de*" → "[NULL]" (3)

is more feasible to implement, though again, there is great ambiguity – a source word may be deleted at any point during the search, with identical target results. Few systems allow this operation in practice. Estimating the likelihood of this operation and correctly identifying the contexts in which it should occur remain challenging problems.

Hierarchical systems, such as (Chiang, 2005) in principle have the capacity to learn insertions and deletions grounded by minimal lexical cues. However, the extracted rules use a single non-terminal. Hence, to avoid explosive ambiguity, they are constrained to contain at least one aligned pair of words. This restriction successfully limits computational complexity at a cost of generalization power.

Syntax-based approaches provide fertile context for grounding insertions and deletions. Often we may draw a strong correspondence between function words in one language and syntactic constructions in another. For instance, the syntactic approach of Marcu et al. (2006) can learn unlexicalized rules that insert function words in isolation, such as:

NP(NN:x0 NN:x1) → x1 *de* x0 (4)

However, as discussed in (Wang, Knight & Marcu, 2007), joint modeling of structure and lexical choice can exacerbate data sparsity, a problem that they attempt to address by tree binarization. Nevertheless, as we show below, unlexicalized structural transformation rules such as (1) and (4) that allow for insertion of isolated function words, are essential for good quality translation of truly out-of-domain test data.

In the treelet translation approach (Menezes & Quirk, 2007), lexical choice and syntactic re-ordering are modeled separately using lexicalized treelets and unlexicalized order templates. We discuss this approach in more detail in Section 4. In Section 5, we describe how we extend this approach to allow for structural insertion and deletion, without the need for content word anchors.

3 Related Work

There is surprisingly little prior work in this area. We previously (Menezes & Quirk, 2005) explored the use of deletion operations such as (3) above, but these were not grounded in any syntactic context, and the estimation was somewhat heuristic[1].

The tuple translation model of Crego et al. (2005), a joint model over source and target translations, also provides a means of deleting words. In training, sentence pairs such as "*nombre de archivo*" / "*file name*" are first word aligned, then minimal bilingual tuples are identified, such as "*nombre* / *name*", "*de* / NULL" and "*archivo* / *file*". The tuples may involve deletion of words by allowing an empty target side, but do not allow insertion tuples with an empty source side. These inserted words are bound to an adjacent neighbor. An n-gram model is trained over the tuple sequences. As a result, deletion probabilities have the desirable property of being conditioned on adjacent context, yet this context is heavily lexicalized, therefore unlikely to generalize well.

More recently, Li et. al. (2008) describe three models for handling "single word deletion" (they discuss, but do not address, word insertion). The first model uses a fixed probability of deletion

[1] We assigned channel probabilities based on the sum of the Model1 probability of the source word being aligned to NULL or one of a list of "garbage collector" words. This exploits the property of Model1 that certain high-frequency words tend to act as "garbage collectors" for words that should remain unaligned.

P(NULL), independent of the source word, estimated by counting null alignments in the training corpus. The second model estimates a deletion probability per-word, P(NULL|w), also directly from the aligned corpus, and the third model trains an SVM to predict the probability of deletion given source language context (neighboring and dependency tree-adjacent words and parts-of-speech). All three models give large gains of 1.5% BLEU or more on Chinese-English translation. It is interesting to note that the more sophisticated models provide a relatively small improvement over the simplest model in-domain, and no benefit out-of-domain.

4 Dependency treelet translation

As a baseline, we use the treelet translation approach (which we previously described in Menezes & Quirk, 2007), a linguistically syntax-based system leveraging a source parser. It first unifies lexicalized treelets and unlexicalized templates to construct a sentence-specific set of synchronous rewrite rules. It then finds the highest scoring derivation according to a linear combination of models. We briefly review this system before describing our current extension.

4.1 The treelet translation model

Sentence-specific rewrite rules are constructed by unifying information from three sources: a dependency parse of the input sentence, a set of treelet translation pairs, and a set of unlexicalized order templates. Dependency parses are represented as trees: each node has a lexical label and a part of speech, as well as ordered lists of pre- and post-modifiers.

A *treelet* represents a connected subgraph of a dependency tree; *treelet translation pairs* consist of source and target treelets and a node alignment. This alignment is represented by indices: each node is annotated with an integer alignment index. A source node and a target node are aligned *iff* they have the same alignment index. For instance:

$$((\mathit{old}_1/\mathrm{JJ})\ \mathit{man}_2/\mathrm{NN}) \rightarrow (\mathit{hombre}_2\ (\mathit{viejo}_1)) \quad (5)$$
$$(\mathit{man}_1/\mathrm{NN}) \rightarrow (\mathit{hombre}_1) \quad (6)$$

Order templates are unlexicalized transduction rules that describe the reorderings, insertions and deletions associated with a single group of nodes that are aligned together. For instance:

$$((\mathrm{x0}{:}\star/\mathrm{DT})\ (\mathrm{x1}{:}\star/\mathrm{JJ})\ \star_1/\mathrm{NN}) \rightarrow ((\mathrm{x0})\ \star_1\ (\mathrm{x1})) \quad (7)$$
$$((\mathrm{x0}{:}\star/\mathrm{DT})\ (\mathrm{x1}{:}\star/\mathrm{JJ})\ \star_1/\mathrm{NN}) \rightarrow ((\mathrm{x0})\ (\mathrm{x1})\ \star_1) \quad (8)$$
$$((\mathrm{x0}{:}\star/\mathrm{DT})\ \star_1/\mathrm{NN}) \rightarrow ((\mathrm{x0})\ \star_1) \quad (9)$$
$$((\mathrm{x0}{:}\star/\mathrm{RB})\ \star_1/\mathrm{JJ}) \rightarrow ((\mathrm{x0})\ \star_1) \quad (10)$$

Each node is either a placeholder or a variable. Placeholders, such as $\star_1$/NN on the source side or $\star_1$ on the target side, have alignment indices and constraints on their parts-of-speech on the source side, but are unconstrained lexically (represented by the $\star$). These unify at translation time with lexicalized treelet nodes with matching parts-of-speech and alignment.

Variables, such as x0:$\star$/DT on the source side and x0:$\star$ on the target side, also have parts-of-speech constraints on the source side. Variables are used to indicate where rewrite rules are recursively applied to translate subtrees. Thus each variable label such as x0, must occur exactly once on each side.

In effect, a template specifies how all the children of a given source node are reordered during translation. If translation were a word-replacement task, then templates would be just simple, single-level tree transducers. However, in the presence of one-to-many and many-to-one translations and unaligned words, templates may span multiple levels in the tree.

As an example, order template (7) indicates that an NN with two pre-modifying subtrees headed by DT and JJ may be translated by using a single word translation of the NN, placing the translation of the DT subtree as a pre-modifier, and placing the translation of the JJ subtree as a post-modifier. As discussed below, this template can unify with the treelet (6) to produce the following rewrite rule:

$$((\mathrm{x0{:}DT})\ (\mathrm{x1{:}JJ})\ \mathit{man}/\mathrm{NN}) \rightarrow ((\mathrm{x0})\ \mathit{hombre}\ (\mathrm{x1})) \quad (11)$$

Matching: A treelet translation pair matches an input parse *iff* there is a unique correspondence between the source side of the treelet pair and a connected subgraph of the input parse.

An order template matches an input parse *iff* there is a unique correspondence between the source side of the template and the input parse, with the additional restriction that all children of input nodes that correspond to placeholder template nodes must be included in the correspondence. For instance, order template (7) matches the parse

((*the*/DT) (*young*/JJ) *colt*/NN) (12)

but not the parse

((*the*/DT) (*old*/JJ) (*grey*/JJ) *mare*/NN) (13)

Finally, an order template matches a treelet translation pair at a given node *iff*, on both source and target sides, there is a correspondence between the treelet translation nodes and template nodes that is consistent with their tree structure and alignments. Furthermore, all placeholder nodes in the template must correspond to some treelet node.

Constructing a sentence-specific rewrite rule is then a process of unifying each treelet with a matching combination of order templates with respect to an input parse. Each treelet node must be unified with one and only one order template placeholder node. Unifying under these constraints produces a rewrite rule that has a one-to-one correspondence between variables in source and target. For instance, given the input parse:

((*the*/DT) ((very/RB) *old*/JJ) *man*/NN) (14)

we can create a rewrite rule from the treelet translation pair (5) by unifying it with the order template (7), which matches at the node *man* and its descendents, and template (10), which matches at the node *old*, to produce the following sentence-specific rewrite rule:

((*the*/DT) ((x1: ⋆/RB) *old*/JJ) *man*/NN) →
((*el*) *hombre* ((x1) *viejo*)) (15)

Note that by using different combinations of order templates, a single treelet can produce multiple rewrite rules. Also, note how treelet translation pairs capture contextual lexical translations but are underspecified with respect to ordering, while order templates separately capture arbitrary reordering phenomena yet are underspecified lexically. Keeping lexical and ordering information orthogonal until runtime allows for the production of novel transduction rules never actually seen in the training corpus, leading to improved generalization power.

Decoding: Given a set of sentence-specific rewrite rules, a standard beam search algorithm is used to find the highest scoring derivation. Derivations are scored according to a linear combination of models.

4.2 Training

The process of extracting treelet translation pairs and order templates begins with parallel sentences. First, the sentence pairs are word segmented on both sides, and the source language sentences are parsed. Next, the sentence pairs are word aligned and the alignments are used to project a target language dependency tree.

Treelet extraction: From each sentence pair S, T with the alignment relation $\sim$, a treelet translation pair consisting of the source treelet $\mathbf{s} \subseteq S$ and the target treelet $\mathbf{t} \subseteq T$ is extracted *iff*:

(1) There exist $s \in \mathbf{s}$ and $t \in \mathbf{t}$ such that $s \sim t$.
(2) For all $s \in S$, and $t \in T$ such that $s \sim t$, $s \in \mathbf{s}$ *iff* $t \in \mathbf{t}$.

Order template extraction is attempted starting from each node S_{root} in the source whose parent is not also aligned to the same target word(s). We identify T_{root}, the highest target node aligned to S_{root}. We initialize the sets S_0 as $\{S_{root}\}$ and T_0 as $\{T_{root}\}$. We expand S_0 to include all nodes adjacent to some element of S_0 that are (a) unaligned, or (b) aligned to some node in T_0. The converse is applied to T_0. This expansion is repeated until we reach a fixed point. Together, S_0 and T_0 make up the placeholder nodes in the extracted order template. We then create one variable in the order template for each direct child of nodes in S_0 and T_0 that is not already included in the order template. *Iff* there is a one-to-one word alignment correspondence between source and target variables, then a template is extracted. This restriction leads to clean templates, at the cost of excluding all templates involving extraposition.

5 Insertion/deletion order templates

In this paper, we extend our previous work to allow for insertion and deletion of words, by allowing unaligned lexical items as part of the otherwise unlexicalized order templates. Grounding insertions and deletions in templates rather than treelets has two major benefits. First, insertion and deletion can be performed even in the absence of specific lexical context, leading to greater generalization power. Secondly, this increased power is tempered by linguistically informative unlexicalized context. Rather than proposing insertions and deletions in any arbitrary setting, we are guided by specific syntactic phenomena. For instance, when translating English noun compounds into Spanish, we often must include a preposition; this generalization is naturally captured using just parts-of-speech.

The inclusion of lexical items in order templates affects the translation system in only a few places: dependency tree projection, order template extraction, and rewrite rule construction at runtime.

Dependency tree projection: During this step of the baseline treelet system, unaligned words are by default attached low, to the lowest aligned neighbor. Although this worked well in conjunction with the discriminative order model, it prevents unaligned nodes from conditioning on relevant context in order templates. Therefore, we change the default attachment of unaligned nodes to be to the highest aligned neighbor; informal experiments showed that this did not noticeably impact translation quality in the baseline system. For example, consider the source parse and aligned target sentence:

$$((\textit{calibrated}_1/\text{JJ})\ (\textit{camera}_2/\text{NN})\ \textit{file}_3/\text{NN}) \\ \textit{archivo}_3\ \textit{de}_4\ \textit{cámara}_2\ \textit{calibrado}_1 \quad (16)$$

Using the baseline projection algorithm would produce this target dependency tree:

$$(\textit{archivo}_3\ ((\textit{de}_4)\ \textit{cámara}_2)\ (\textit{calibrado}_1)) \quad (17)$$

Instead, we attach unaligned words high:

$$(\textit{archivo}_3\ (\textit{de}_4)\ (\textit{cámara}_2)\ (\textit{calibrado}_1)) \quad (18)$$

Order template extraction: In addition to the purely unlexicalized templates extracted from each training sentence, we also allow templates that include lexical items for each unaligned token. For each point in the original extraction procedure, where S_0 or T_0 contain unaligned nodes, we now extract two templates: The original unlexicalized template, and a new template in which only the unaligned node(s) contain the specific lexical item(s). From the example sentence pair (16), using the projected parse (18) we would extract the following two templates:

$$((\text{x0}{:}\star/\text{JJ})\ (\text{x1}{:}\star/\text{NN})\ \star_1/\text{NN}) \rightarrow \\ (\star_1\ (\star_2)\ (\text{x1})\ (\text{x0})) \quad (19)$$

$$((\text{x0}{:}\star/\text{JJ})\ (\text{x1}{:}\star/\text{NN})\ \star_1/\text{NN}) \rightarrow \\ (\star_1\ (\textit{de}_2)\ (\text{x1})\ (\text{x0})) \quad (20)$$

Template matching and unification: We extend the template matching against the input parse to require that any lexicalized source template nodes match the input exactly. When matching templates to treelet translation pairs, any unaligned treelet nodes must be consistent with the corresponding template node (i.e. the template node must be unlexicalized, or the lexical items must match). On the other hand, lexicalized template nodes do not need to match any treelet nodes -- insertions or deletions may now come from the template alone.

Consider the following example input parse:

$$((\textit{digital}/\text{JJ})\ (\textit{camera}/\text{NN}) \\ (\textit{file}/\text{NN})\ \textit{extension}/\text{NN}) \quad (21)$$

The following treelet translation pair provides a contextual translation for some of the children, including the insertion of one necessary preposition:

$$((\textit{file}_1/\text{NN})\ \textit{extension}_2/\text{NN}) \rightarrow \\ (\textit{extension}_2\ (\textit{de}_3)\ (\textit{archivo}_1)) \quad (22)$$

The following order template can provide relative ordering information between nodes as well as insert the remaining prepositions:

$$((\text{x0}{:}\star/\text{JJ})\ (\text{x1}{:}\star/\text{NN})\ (\text{x2}{:}\star/\text{NN})\ \star_1/\text{NN}) \rightarrow \\ (\star_1\ (\textit{de}_2)\ (\text{x2})\ (\textit{de}_3)\ (\text{x0})\ (\text{x1})) \quad (23)$$

The unification of this template and treelet is somewhat complex: the first inserted *de* is agreed upon by both template and treelet, whereas the second is inserted by the template alone. This results in the following novel rewrite rule:

$$((\text{x0}{:}\star/\text{JJ})\ (\text{x1}{:}\ \star/\text{NN})\ (\textit{file})\ \textit{extension}) \rightarrow \\ (\textit{extension}\ (\textit{de})\ (\textit{archivo})\ (\textit{de})\ (\text{x0})\ (\text{x1})) \quad (24)$$

These relatively minimal changes produce a powerful contextualized model of insertion and deletion.

Parameter estimation: The underlying treelet system includes a template probability estimated by relative frequency. We estimate our lexicalized templates in the same way. However early experiments showed that this feature alone was not enough to allow even common insertions, since the probability of even the most common insertion templates is much lower than that of unlexicalized templates. To improve the modeling capability, we included two additional feature functions: a count of structurally inserted words, and a count of structurally deleted words.

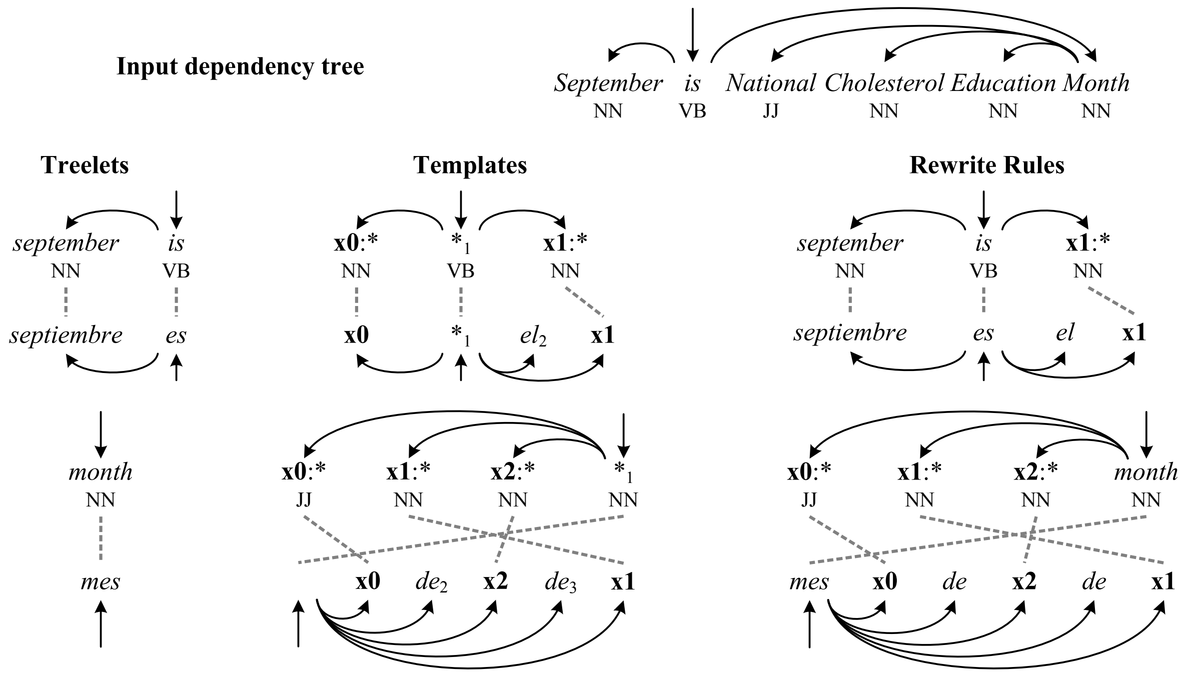


Figure 6.1: Example sentence, matching treelets, structural insertion templates and unified rewrite rules

6 Example

Consider the following English test sentence and corresponding Spanish human translation:

September is National Cholesterol Education Month

Septiembre es el Mes Nacional para la Educación sobre el Colesterol

The baseline treelet system without structural insertions translates this sentence as:

Septiembre es Nacional Colesterol Educación Mes

Not only is the translation missing the appropriate articles and prepositions, but also in their absence, it fails to reorder the content words correctly. Without the missing prepositions, the language model does not show a strong preference among various orderings of "*nacional*" "*colesterol*" "*educación*" and "*mes*".

Using structural insertion templates, the highest scoring translation of the sentence is now:

Septiembre es el Mes Nacional de Educación de colesterol

Although the choice of prepositions is not the same as the reference, the fluency is much improved and the translation is quite understandable. Figure 6.1, lists the structural insertion templates that are used to produce this translation, and shows how they are unified with treelet translation pairs to produce sentence-specific rewrite rules, which are in turn composed during decoding to produce this translation.

7 Experiments

We evaluated the translation quality of the system using the BLEU metric (Papineni et al., 2002). We compared three systems: (a) a standard phrasal system using a decoder based on Pharaoh, (Koehn et al., 2003), (b) A baseline treelet system using unlexicalized order templates and (c) The present work, which adds structural insertion and deletion templates.

7.1 Data

We report results for two language pairs, English-Spanish and English- Japanese. For English-Spanish we use two training sets: (a) the Europarl corpus provided by the NAACL 2006 Statistical Machine Translation workshop (b) a "general-domain" data set that includes a broad spectrum of data such as governmental data, general web data and technical corpora.

For English-Japanese we use only the “general-domain” data set.

	Sentence pairs	Tokens	Phr size	MERT data
Europarl E-S	730K	15M	7	Europarl
General E-S	3.7M	41M	4	Web
General E-J	2.6M	16M	4	Web

Table 7.1 Training data

For English-Spanish we report results using the four test sets listed in Table 7.2. For English-Japanese we use only the web test set. The first two tests are from the 2006 SMT workshop and the newswire test is from the 2008 workshop. The web test sets were selected from a random sampling of English web sites, with target language translations provided by professional translation vendors. All test sets have one reference translation.

	Domain	Sentence pairs
eu-test	Europarl	2000
nc-test	News commentary	1064
News	News wire	2051
Web	General web text	5000

Table 7.2 Test data

7.2 Models

The baseline treelet translation system uses all the models described in Menezes & Quirk (2007), namely:

- Treelet log probabilities, maximum likelihood estimates with absolute discounting.
- Forward and backward lexical weighting, using Model-1 translation log probabilities.
- Trigram language model using modified Kneser-Ney smoothing.
- Word and phrase count feature functions.
- Order template log probabilities, maximum likelihood estimates, absolute discounting.
- Count of artificial *source order templates.*[2]
- Discriminative tree-based order model.

The present work does not use the discriminative tree-based order model[3] but adds:

- Count of structural insertions: This counts only words inserted via templates, not lexical insertions via treelets.
- Count of structural deletions: This counts only words deleted via templates, not lexical deletions via treelets.

The comparison phrasal system was constructed using the same alignments and the heuristic combination described in (Koehn et al., 2003). This system used a standard set of models:

- Direct and inverse log probabilities, both relative frequency and lexical weighting.
- Word count, phrase count.
- Trigram language model log probability.
- Length based distortion model.
- Lexicalized reordering model.

7.3 Training

We parsed the source (English) side of the corpus using NLPWIN, a broad-coverage rule-based parser able to produce syntactic analyses at varying levels of depth (Heidorn, 2000). For the purposes of these experiments, we used a dependency tree output with part-of-speech tags and unstemmed, case-normalized surface words. For word alignment we used a training regimen of five iterations of Model 1, followed by five iterations of a word-dependent HMM model (He, 2007) in both directions. The forward and backward alignments were combined using a dependency tree-based heuristic combination. The word alignments and English dependency tree were used to project a target tree. From the aligned tree pairs we extracted treelet and order template tables.

For the Europarl systems, we use a phrase/treelet size of 7 and train model weights using 2000 sentences of Europarl data. For the “general-domain” systems, we use a phrase/treelet size of 4, and train model weights using 2000 sentences of web data.

For any given corpus, all systems used the same treelet or phrase size (see Table 7.1) and the same trigram language model. Model weights were trained separately for each system, data set and experimental condition, using minimum error rate training to maximize BLEU (Och, 2003).

[2] When no template is compatible with a treelet, the decoder creates an artificial template that preserves source order. This count feature allows MERT to deprecate the use of such templates. This is analogous to the glue rules of Chiang (2005).

[3] In our experiments, we find that the impact of this model is small in the presence of order templates; also, it degrades the overall speed of the decoder.

	% BLEU
Phrasal	13.41
Baseline treelet	15.89
+Deletion only	16.00
+Insertion only	16.16
+Deletion and Insertion	**17.01**

Table 8.1: English-Japanese system comparisons

8 Results and Discussion

Tables 8.1 and 8.4 compare baseline phrasal and treelet systems with systems that use various types of insertion and deletion templates.

English-Japanese: As one might expect, the use of structural insertion and deletion has the greatest impact when translating between languages such as English and Japanese that show significant structural divergence. In this language pair, both insertions and deletions have an impact, for a total gain of 1.1% BLEU over the baseline treelet system, and 3.6% over the phrasal system. To aid our understanding of the system, we tabulated the most commonly inserted and deleted words when translating from English into Japanese in Tables 8.2 and 8.3 respectively. Satisfyingly, most of the insertions and deletions correspond to well-known structural differences between the languages. For instance, in English the thematic role of a noun phrase, such as subject or object, is typically indicated by word order, whereas Japanese uses case markers to express this information. Hence, case markers such as "を" and "は" need to be inserted. Also, when noun compounds are translated, an intervening postposition such as "の" is usually needed. Among the most common deletions are "*the*" and "*a*". This is because Japanese does not have a notion of definiteness. Similarly, pronouns are often dropped in Japanese.

English-Spanish: We note, in Table 8.4 that even between such closely related languages, structural insertions give us noticeable improvements over the baseline treelet system. On the smaller Europarl training corpus the improvements range from 0.5% to 1.1% BLEU. On the larger training corpus we find that for the more in-domain governmental[4] and news test sets, the effect is smaller or even slightly negative, but on the very broad web test set we still see an improvement of about 0.7% BLEU.

[4] The "general domain" training corpus is a superset of the Europarl training set, therefore, the Europarl tests sets are "in-domain" in both cases.

Word	Count	%age	Type
の	2844	42%	Postposition
を	1637	24%	Postposition/case marker
は	630	9.3%	Postposition/case marker
、	517	7.6%	Punctuation
に	476	7.0%	Postposition
する	266	3.9%	Light verb
で	101	1.5%	Postposition
が	68	1.0%	Postposition
して	27	0.40%	Light verb
。	26	0.38%	Punctuation
か	19	0.28%	Question marker

Table 8.2: E-J: Most commonly inserted words

Word	Count	%age	Type
the	875	59%	Definite article
-	159	11%	Punctuation
a	113	7.7%	Indefinite article
you	53	3.6%	Pronoun
it	53	3.6%	Pronoun
that	26	1.8%	Conjunction, Pronoun
"	23	1.6%	Punctuation
in	16	1.1%	Preposition
.	10	0.68%	Punctuation
's	10	0.68%	Possessive
I	9	0.61%	Pronoun

Table 8.3: E-J: Most commonly deleted words

As one might expect, as the training data size increases, the generalization power of structural insertion and deletions becomes less important when translating *in-domain* text, as more insertions and deletions can be handled lexically. Nevertheless, the web test results indicate that if one hopes to handle truly general input the need for structural generalizations remains.

Unlike in English-Japanese, when translating from English to Spanish, structural deletions are less helpful. Used in isolation or in combination with insertion templates they have a slightly negative and/or insignificant impact in all cases. We hypothesize that when translating *from* English *into* Spanish, more words need to be inserted than deleted. Conversely, when translating in the reverse direction, deletion templates may play a bigger role. We were unable to test the reverse direction because our syntax-based systems depend on a source language parser. In future work we hope to address this.

		EU-devtest	EU-test	NC-test	Newswire	Web test
EUROPARL E-S						
	Phrasal	27.9	28.5	24.7	17.7	17.0
	Baseline treelet	27.65	28.38	27.00	18.46	18.71
	+Deletion only	27.66	28.39	26.97	18.46	18.64
	+Insertion only	28.23	28.93	**28.10**	**19.08**	**19.43**
	+Deletion and Insertion	**28.27**	**29.08**	27.82	18.98	19.19
GENERAL E-S						
	Phrasal	28.79	29.19	29.45	21.12	27.91
	Baseline treelet	28.67	29.33	32.49	**21.90**	27.42
	+Deletion only	28.67	29.27	32.25	21.69	27.47
	+Insertion only	**28.90**	**29.70**	32.53	21.84	**28.30**
	+Deletion and Insertion	28.34	29.41	**32.66**	21.70	27.95

Table 8.4: English-Spanish system comparisons, %BLEU

In table 8.5 and 8.6, we list the words most commonly inserted and deleted when translating the web test using the general English-Spanish system. As in English-Japanese, we find that the insertions are what one would expect on linguistic grounds. However, deletions are used much less frequently than insertions and also much less frequently than they are in English-Japanese. Only 53 words are structurally deleted in the 5000 sentence test set, as opposed to 4728 structural insertions. Furthermore, the most common deletion is of quotation marks, which is incorrect in most cases, even though such deletion is evidenced in the training corpus[5].

On the other hand, the next most common deletions "*I*" and "*it*" are linguistically well grounded, since Spanish often drops pronouns.

de	3509	74%	Preposition
la	555	12%	Determiner
el	250	5.3%	Determiner
se	77	1.6%	Reflexive pronoun
que	63	1.3%	Relative pronoun
los	63	1.3%	Determiner
del	57	1.2%	Preposition+Determiner
,	42	0.89%	Punctuation
a	30	0.63%	Preposition
en	21	0.44%	Preposition
lo	9	0.19%	Pronoun
las	6	0.13%	Determiner

Table 8.5: E-S: Most commonly inserted words

"	38	72%	Punctuation
I	5	9.4%	Pronoun
it	2	3.8%	Pronoun
,	2	3.8%	Punctuation
-	2	3.8%	Punctuation

Table 8.6: E-S: Most commonly deleted words

9 Conclusions and Future Work

We have presented an extension of the treelet translation method to include order templates with structural insertion and deletion, which improves translation quality under a variety of scenarios, particularly between structurally divergent languages. Even between closely related languages, these operations significantly improve the generalizability of the system, providing benefit when handling out-of-domain test data.

Our experiments shed light on a little-studied area of MT, but one that is nonetheless crucial for high quality broad domain translation. Our results affirm the importance of structural insertions, in particular, when translating from English into other languages, and the importance of both insertions and deletions when translating between divergent languages. In future, we hope to study translations from other languages into English to study the role of deletions in such cases.

[5] In many parallel corpora, quotes are not consistently preserved between source and target languages.

References

Chiang, David. A hierarchical phrase-based model for statistical machine translation. ACL 2005.

Crego, Josep, José Mariño and Adrià de Gispert. Reordered search and tuple unfolding for Ngram-based SMT. MT Summit 2005.

He, Xiaodong. Using Word Dependent Transition Models in HMM based Word Alignment for Statistical Machine Translation. Workshop on Statistical Machine Translation, 2007

Heidorn, George. “Intelligent writing assistance”. In Dale et al. Handbook of Natural Language Processing, Marcel Dekker. 2000

Koehn, Philipp, Franz Josef Och, and Daniel Marcu. Statistical phrase based translation. NAACL 2003.

Chi-Ho Li, Dongdong Zhang, Mu Li, Ming Zhou, Hailei Zhang. An Empirical Study in SourceWord Deletion for Phrase-based Statistical Machine Translation. Workshop on Statistical Machine Translation, 2008

Marcu, Daniel, Wei Wang, Abdessamad Echihabi, and Kevin Knight. SPMT: Statistical Machine Translation with Syntactified Target Language Phrases. EMNLP-2006.

Menezes, Arul, and Chris Quirk. Microsoft Research Treelet translation system: IWSLT evaluation. International Workshop on Spoken Language Translation, 2005

Menezes, Arul, and Chris Quirk. Using Dependency Order Templates to Improve Generality in Translation. Workshop on Statistical Machine Translation, 2007

Och, Franz Josef. Minimum error rate training in statistical machine translation. ACL 2003.

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

Wang, Wei, Kevin Knight and Daniel Marcu. Binarizing Syntax Trees to Improve Syntax-Based Machine Translation Accuracy. EMNLP-CoNLL, 2007

Predicting Success in Machine Translation

Alexandra Birch **Miles Osborne** **Philipp Koehn**
a.c.birch-mayne@sms.ed.ac.uk miles@inf.ed.ac.uk pkoehn@inf.ed.ac.uk
School of Informatics
University of Edinburgh
10 Crichton Street
Edinburgh, EH8 9AB, UK

Abstract

The performance of machine translation systems varies greatly depending on the source and target languages involved. Determining the contribution of different characteristics of language pairs on system performance is key to knowing what aspects of machine translation to improve and which are irrelevant. This paper investigates the effect of different explanatory variables on the performance of a phrase-based system for 110 European language pairs. We show that three factors are strong predictors of performance in isolation: the amount of reordering, the morphological complexity of the target language and the historical relatedness of the two languages. Together, these factors contribute 75% to the variability of the performance of the system.

1 Introduction

Statistical machine translation (SMT) has improved over the last decade of intensive research, but for some language pairs, translation quality is still low. Certain systematic differences between languages can be used to predict this. Many researchers have speculated on the reasons why machine translation is hard. However, there has never been, to our knowledge, an analysis of what the actual contribution of different aspects of language pairs is to translation performance. This understanding of where the difficulties lie will allow researchers to know where to most gainfully direct their efforts to improving the current models of machine translation.

Many of the challenges of SMT were first outlined by Brown et al. (1993). The original IBM Models were broken down into separate translation and distortion models, recognizing the importance of word order differences in modeling translation. Brown et al. also highlighted the importance of modeling morphology, both for reducing sparse counts and improving parameter estimation and for the correct production of translated forms. We see these two factors, reordering and morphology, as fundamental to the quality of machine translation output, and we would like to quantify their impact on system performance.

It is not sufficient, however, to analyze the morphological complexity of the source and target languages. It is also very important to know how similar the morphology is between the two languages, as two languages which are morphologically complex in very similar ways, could be relatively easy to translate. Therefore, we also include a measure of the family relatedness of languages in our analysis.

The impact of these factors on translation is measured by using linear regression models. We perform the analysis with data from 110 different language pairs drawn from the Europarl project (Koehn, 2005). This contains parallel data for the 11 official language pairs of the European Union, providing a rich variety of different language characteristics for our experiments. Many research papers report results on only one or two languages pairs. By analyzing so many language pairs, we are able to provide a much wider perspective on the challenges facing machine translation. This analysis is important as it provides very strong motivation for further research.

The findings of this paper are as follows: (1) each of the main effects, reordering, target language complexity and language relatedness, is a highly significant predictor of translation performance, (2) individually these effects account for just over a third of the variation of the BLEU score, (3) taken together, they account for 75% of the variation of the BLEU score, (4) when removing Finnish results as outliers, reordering explains the most variation, and finally (4) the morphological complexity of the source language is uncorrelated with performance, which suggests that any difficulties that arise with sparse counts are insignificant under the experimental conditions outlined in this paper.

2 Europarl

In order to analyze the influence of different language pair characteristics on translation performance, we need access to a large variety of comparable parallel corpora. A good data source for this is the Europarl Corpus (Koehn, 2005). It is a collection of the proceedings of the European Parliament, dating back to 1996. Version 3 of the corpus consists of up to 44 million words for each of the 11 official languages of the European Union: Danish (da), German (de), Greek (el), English (en), Spanish (es), Finnish (fi), French (fr), Italian (it), Dutch (nl), Portuguese (pt), and Swedish (sv).

In trying to determine the effect of properties of the languages involved in translation performance, it is very important that other variables be kept constant. Using Europarl, the size of the training data for the different language pairs is very similar, and there are no domain differences as all sentences are roughly trained on translations of the same data.

3 Morphological Complexity

The morphological complexity of the language pairs involved in translation is widely recognized as one of the factors influencing translation performance. However, most statistical translation systems treat different inflected forms of the same lemma as completely independent of one another. This can result in sparse statistics and poorly estimated models. Furthermore, different variations of the lemma may result in crucial differences in meaning that affect the quality of the translation.

Work on improving MT systems' treatment of morphology has focussed on either reducing word forms to lemmas to reduce sparsity (Goldwater and McClosky, 2005; Talbot and Osborne, 2006) or including morphological information in decoding (Dyer, 2007).

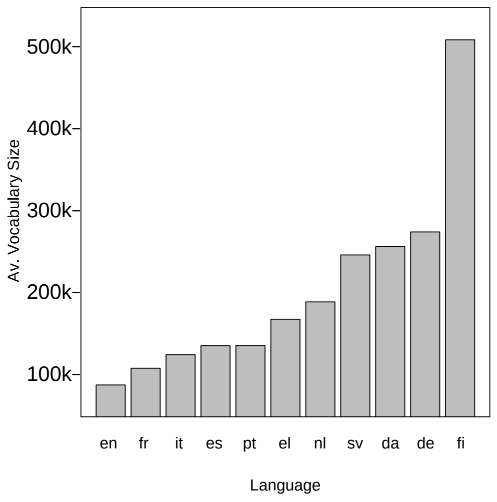


Figure 1. Average vocabulary size for each language.

Although there is a significant amount of research into improving the treatment of morphology, in this paper we aim to discover the effect that different levels of morphology have on translation. We measure the amount of morphological complexity that exists in both languages and then relate this to translation performance.

Some languages seem to be intuitively more complex than others, for instance Finnish appears more complex than English. There is, however, no obvious way of measuring this complexity. One method of measuring complexity is by choosing a number of hand-picked, intuitive properties called *complexity indicators* (Bickel and Nichols, 2005) and then to count their occurrences. Examples of morphological complexity indicators could be the number of inflectional categories or morpheme types in a typical sentence. This method suffers from the major drawback of finding a principled way of choosing which of the many possible linguistic properties should be included in the list of indicators.

A simple alternative employed by Koehn (2005) is to use vocabulary size as a measure of morphological complexity. Vocabulary size is strongly influenced by the number of word forms affected by number, case, tense etc. and it is also affected by the number of agglutinations in the language. The complexity of the morphology of languages can therefore be approached by looking at vocabulary size.

Figure 1 shows the vocabulary size for all relevant languages. Each language pair has a slightly different parallel corpus, and so the size of the vocabularies for each language needs to be averaged. You can see that the size of the Finnish vocabulary is about six times larger (510,632 words) than the English vocabulary size (88,880 words). The reason for the large vocabulary size is that Finnish is characterized by a rich inflectional morphology, and it is typologically classified as an agglutinative-fusional language. As a result, words are often polymorphemic, and become remarkably long.

4 Language Relatedness

The morphological complexity of each language in isolation could be misleading. Large differences in morphology between two languages could be more relevant to translation performance than a complex morphology that is very similar in both languages. Languages which are closely related could share morphological forms which might be captured reasonably well in translation models. We include a measure of language relatedness in our analyses to take this into account.

Comparative linguistics is a field of linguistics which aims to determine the historical relatedness of languages. Lexicostatistics, developed by Morris Swadesh in the 1950s (Swadesh, 1955), is an approach to comparative linguistics that is appropriate for our purposes because it results in a quantitative measure of relatedness by comparing lists of lexical cognates.

The lexicostatistic percentages are extracted as follows. First, a list of universal culture-free meanings are generated. Words are then collected for these meanings for each language under consideration. Lists for particular purposes have been generated. For example, we use the data from Dyen et al. (1992) who developed a list of 200 meanings for 84 Indo-European languages. Cognacy decisions are then made by a trained linguist. For each pair of lists the cognacy of a form can be positive, negative or indeterminate. Finally, the lexicostatistic percentage is calculated. This percentage is related to the proportion of meanings for a particular language pair that are cognates, i.e. relative to the total without indeterminacy. Factors such as borrowing, tradition and taboo can skew the results.

Language	“animal”	“black”
French	animal	noir
Italian	animale	nero
Spanish	animal	negro
English	animal	black
German	tier	schwarz
Swedish	djur	svart
Danish	dyr	sort
Dutch	dier	zwart

Table 1. An example from the (Dyen et al., 1992) cognate list.

A portion of the Dyen et al. (1992) data set is shown in Table 1 as an example. From this data a trained linguist would calculate the relatedness of French, Italian and Spanish as 100% because their words for “animal” and “black” are cognates. The Romance languages share one cognate with English, “animal” but not “black”, which means that the lexicostatistic percentage here would be 50%, and no cognates with the rest of the languages, 0%.

We use the Dyen lexicostatistic percentages as our measure of language relatedness or similarity for all bidirectional language pairs except for Finnish, for which there is not data. Finnish is a Finno-Ugric language and is not part of the Indo-European language family and is therefore not included in the Dyen results. We were not able to recreate the conditions of this study to generate the data for Finnish - expert linguists with knowledge of all the languages would be required. Excluding Finnish would have been a shame as it is an interesting language to look at, however we took care to confirm which effects found in this paper still held when excluding Finnish. Not being part of the Indo-European languages means that its historical similarity with our other languages is very low. For example, English would be more closely related to Hindu than to Finnish. We therefore assume that Finnish has zero similarity with the other languages in the set.

Figure 2 shows the symmetric matrix of language relatedness, where the width of the square is proportional to the value of relatedness. Finnish is the language which is least related to the other languages and has a relatedness score of 0%. Spanish-Portuguese is the most related language pair with a

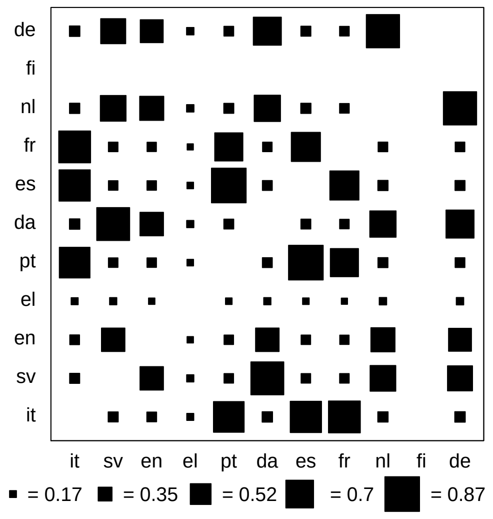


Figure 2. Language relatedness - the width of the squares indicates the lexicostatical relatedness.

score of 0.87%.

A measure of family relatedness should improve our understanding of the relationship between morphological complexity and translation performance.

5 Reordering

Reordering refers to differences in word order that occur in a parallel corpus and the amount of reordering affects the performance of a machine translation system. In order to determine how much it affects performance, we first need to measure it.

5.1 Extracting Reorderings

Reordering is largely driven by syntactic differences between languages and can involve complex rearrangements between nodes in synchronous trees. Modeling reordering exactly would require a synchronous tree-substitution grammar. This representation would be sparse and heterogeneous, limiting its usefulness as a basis for analysis. We make an important simplifying assumption in order for the detection and extraction of reordering data to be tractable and useful. We assume that reordering is a binary process occurring between two blocks that are adjacent in the source. This is similar to the ITG constraint (Wu, 1997), however our reorderings are not dependent on a synchronous grammar or a derivation which covers the sentences. There are also similarities with the Human-Targeted Translation Edit Rate metric (HTER) (Snover et al., 2006) which attempts to find the minimum number of human edits to correct a hypothesis, and admits moving blocks of words, however our algorithm is automatic and does not consider inserts or deletes.

Before describing the extraction of reorderings we need to define some concepts. We define a *block* A as consisting of a source span, $A_{\bar{s}}$, which contains the positions from A_{smin} to A_{smax} and is aligned to a set of target words. The minimum and maximum positions (A_{tmin} and A_{tmax}) of the aligned target words mark the block's target span, $A_{\bar{t}}$.

A reordering r consists of the two blocks r_A and r_B, which are adjacent in the source and where the relative order of the blocks in the source is reversed in the target. More formally:

$$r_{A_{\bar{s}}} < r_{B_{\bar{s}}}, \quad r_{A_{\bar{t}}} > r_{B_{\bar{t}}}, \quad r_{A_{smax}} = r_{B_{smin}} - 1$$

A consistent block means that between A_{tmin} and A_{tmax} there are no target word positions aligned to source words outside of the block's source span $A_{\bar{s}}$. A reordering is consistent if the block projected from $r_{A_{smin}}$ to $r_{B_{smax}}$ is consistent.

The following algorithm detects reorderings and determines the dimensions of the blocks involved. We step through all the source words, and if a word is reordered in the target with respect to the previous source word, then a reordering is said to have occurred. These two words are initially defined as the blocks A and B. Then the algorithm attempts to grow block A from this point towards the source starting position, while the target span of A is greater than that of block B, and the new block A is consistent. Finally it attempts to grow block B towards the source end position, while the target span of B is less than that of A and the new reordering is inconsistent.

See Figure 3 for an example of a sentence pair with two reorderings. Initially a reordering is detected between the Chinese words aligned to "from" and "late". The block A is grown from "late" to include the whole phrase pair "late last night". Then the block B is grown from "from" to include "Beijing" and stops because the reordering is then consistent. The next reordering is detected between "arrived in" and "Beijing". We can see that block A attempts to grow as large a block as possible and block

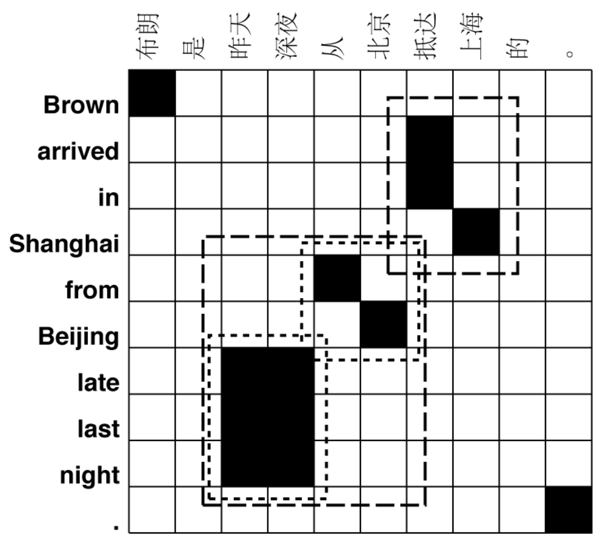


Figure 3. A sentence pair from the test corpus, with its alignment. Two reorderings are shown with two different dash styles.

B attempts to grow the smallest block possible. The reorderings thus extracted would be comparable to those of a right-branching ITG with inversions. This allows for syntactically plausible embedded reorderings. This algorithm has the worst case complexity of $O(\frac{n^2}{2})$ when the words in the target occur in the reverse order to the words in the source.

5.2 Measuring Reordering

Our reordering extraction technique allows us to analyze reorderings in corpora according to the distribution of reordering widths. In order to facilitate the comparison of different corpora, we combine statistics about individual reorderings into a sentence level metric which is then averaged over a corpus.

$$RQuantity = \frac{\sum_{r \in R} |r_{A_{\bar{s}}}| + |r_{B_{\bar{s}}}|}{I}$$

where R is the set of reorderings for a sentence, I is the source sentence length, A and B are the two blocks involved in the reordering, and $|r_{A_{\bar{s}}}|$ is the size or span of block A on the source side. RQuantity is thus the sum of the spans of all the reordering blocks on the source side, normalized by the length of the source sentence.

	RQuantity
Europarl, auto align	0.620
WMT06 test, auto align	0.647
WMT06 test, manual align	0.668

Table 2. The reordering quantity for the different reordering corpora for DE-EN.

5.3 Automatic Alignments

Reorderings extracted from manually aligned data can be reliably assumed to be correct. The only exception to this is that embedded reorderings are always right branching and these might contradict syntactic structure. In this paper, however, we use alignments that are automatically extracted from the training corpus using GIZA++. Automatic alignments could give very different reordering results. In order to justify using reordering data extracted from automatic alignments, we must show that they are similar enough to gold standard alignments to be useful as a measure of reordering.

5.3.1 Experimental Design

We select the German-English language pair because it has a reasonably high level of reordering. A manually aligned German-English corpus was provided by Chris Callison-Burch and consists of the first 220 sentences of test data from the 2006 ACL Workshop on Machine Translation (WMT06) test set. This test set is from a held out portion of the Europarl corpus.

The automatic alignments were extracted by appending the manually aligned sentences on to the respective Europarl v3 corpora and aligning them using GIZA++ (Och and Ney, 2003) and the grow-final-diag algorithm (Koehn et al., 2003).

5.3.2 Results

In order to use automatic alignments to extract reordering statistics, we need to show that reorderings from automatic alignments are comparable to those from manual alignments.

We first look at global reordering statistics and then we look in more detail at the reordering distribution of the corpora. Table 2 shows the amount of reordering in the WMT06 test corpora, with both manual and automatic alignments, and in the automatically aligned Europarl DE-EN parallel corpus.

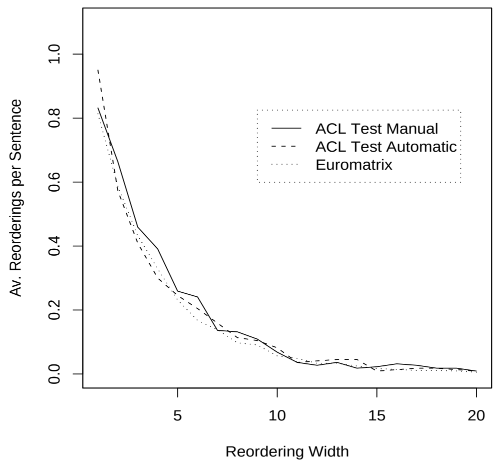


Figure 4. Average number of reorderings per sentence mapped against the total width of the reorderings for DE-EN.

We can see that all three corpora show a similar amount of reordering.

Figure 4 shows that the distribution of reorderings between the three corpora is also very similar. These results provide evidence to support our use of automatic reorderings in lieu of manually annotated alignments. Firstly, they show that our WMT06 test corpus is very similar to the Europarl data, which means that any conclusions that we reach using the WMT06 test corpus will be valid for the Europarl data. Secondly, they show that the reordering behavior of this corpus is very similar when looking at automatic vs. manual alignments.

Although differences between the reorderings detected in the manually and automatically aligned German-English corpora are minor, there we accept that there could be a language pair whose real reordering amount is very different to the expected amount given by the automatic alignments. A particular language pair could have alignments that are very unsuited to the stochastic assumptions of the IBM or HMM alignment models. However, manually aligning 110 language pairs is impractical.

5.4 Amount of reordering for the matrix

Extracting the amount of reordering for each of the 110 language pairs in the matrix required a sampling approach. We randomly extracted a subset of 2000 sentences from each of the parallel training corpora. From this subset we then extracted the average RQuantity.

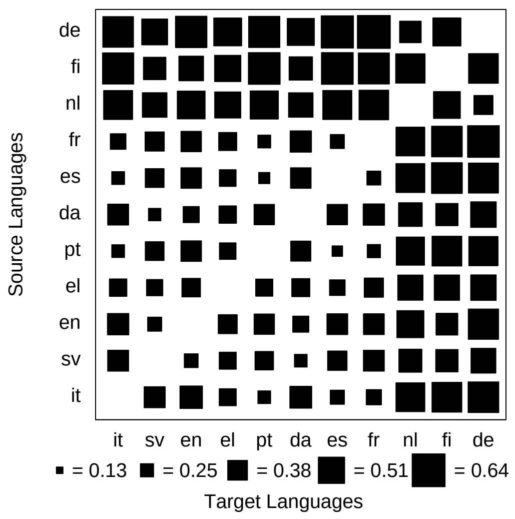


Figure 5. Reordering amount - the width of the squares indicates the amount of reordering or RQuantity.

In Figure 5 the amount of reordering for each of the language pairs is proportional to the width of the relevant square. Note that the matrix is not quite symmetrical - reordering results differ depending on which language is chosen to measure the reordering span. The lowest reordering scores are generally for languages in the same language group (like Portuguese-Spanish, 0.20, and Danish-Swedish, 0.24) and the highest for languages from different groups (like German-French, 0.64, and Finnish-Spanish, 0.61).

5.5 Language similarity and reordering

In this paper we use linear regression models to determine the correlation and significance of various explanatory variables with the dependent variable, the BLEU score. Ideally the explanatory variables involved should be independent of each other, however the amount of reordering in a parallel corpus could easily be influenced by family relatedness. We investigate the correlation between these variables.

Figure 6 shows the plot of the reordering amount against language similarity. The regression is highly significant and has an R^2 of 0.2347. This means that reordering is correlated with language similarity and that 23% of reordering can be explained by language similarity.

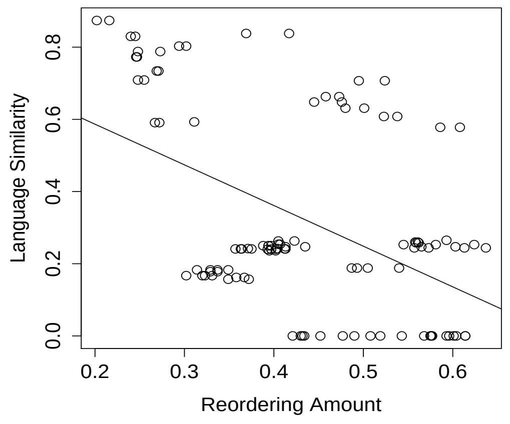


Figure 6. Reordering compared to language similarity with regression.

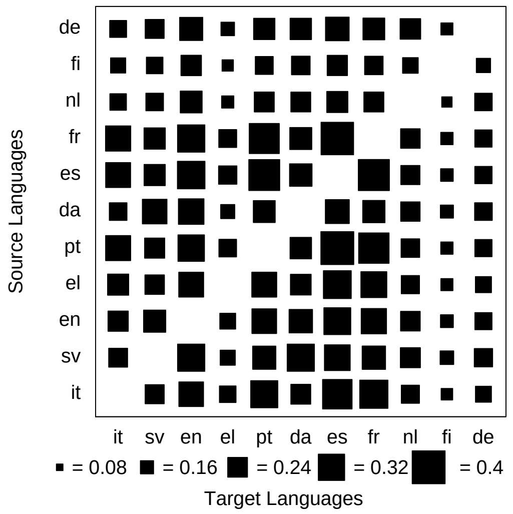


Figure 7. System performance - the width of the squares indicates the system performance in terms of the BLEU score.

6 Experimental Design

We used the phrase-based model Moses (Koehn et al., 2007) for the experiments with all the standard settings, including a lexicalized reordering model, and a 5-gram language model. Tests were run on the ACL WSMT 2008 test set (Callison-Burch et al., 2008).

6.1 Evaluation of Translation Performance

We use the BLEU score (Papineni et al., 2002) to evaluate our systems. While the role of BLEU in machine translation evaluation is a much discussed topic, it is generally assumed to be a adequate metric for comparing systems of the same type.

Figure 7 shows the BLEU score results for the matrix. Comparing this figure to Figure 5 there seems to be a clear negative correlation between reordering amount and translation performance.

6.2 Regression Analysis

We perform multiple linear regression analyses using measures of morphological complexity, language relatedness and reordering amount as our independent variables. The dependent variable is the translation performance metric, the BLEU score.

We then use a t-test to determine whether the coefficients for the independent variables are reliably different from zero. We also test how well the model explains the data using an R^2 test. The two-tailed significance levels of coefficients and R^2 are also given where * means $p < 0.05$, ** means $p < 0.01$, and *** means $p < 0.001$.

Explanatory Variable	Coefficient	
Target Vocab. Size	-3.885	***
Language Similarity	3.274	***
Reordering Amount	-1.883	***
Target Vocab. Size^2	1.017	***
Language Similarity^2	-1.858	**
Interaction: Reord/Sim	-1.4536	***

Table 3. The impact of the various explanatory features on the BLEU score via their coefficients in the minimal adequate model.

7 Results

7.1 Combined Model

The first question we are interested in answering is which factors contribute most and how they interact. We fit a multiple regression model to the data. The source vocabulary size has no significant effect on the outcome. All explanatory variable vectors were normalized to be more comparable.

In Table 3 we can see the relative contribution of the different features to the model. Source vocabulary size did not contribute significantly to the explanatory power of this multiple regression model and was therefore not included. The fraction of the variance explained by the model, or its goodness of fit, the R^2, is 0.750 which means that 75% of the variation in BLEU can be explained by these three factors. The interaction of reordering amount and language relatedness is the product of the values of these two features, and in itself it is an important explanatory feature.

To make sure that our regression is valid, we need to consider the special case of Finnish. Data points where Finnish is the target language are outliers. Finnish has the lowest language similarity with all other languages, and the largest vocabulary size. It also has very high amounts of reordering, and the lowest BLEU scores when it is the target language. The multiple regression of Table 3 where Finnish as the source and target language is excluded, shows that all the effects are still very significant, with the model's R^2 dropping only slightly to 0.68.

The coefficients of the variables in the multiple regression model have only limited usefulness as a measure of the impact of the explanatory variables in the model. One important factor to consider is that if the explanatory variables are highly correlated, then the values of the coefficients are unstable. The model could attribute more importance to one or the other variable without changing the overall fit of the model. This is the problem of multicollinearity. Our explanatory variables are all correlated, but a large amount of this correlation can be explained by looking at language pairs with Finnish as the target language. Excluding these data points, only language relatedness and reordering amount are still correlated, see Section 5.5 for more details.

7.2 Contribution in isolation

In order to establish the relative contribution of variables, we isolate their impact on the BLEU score by modeling them in separate linear regression models.

Figure 8 shows a simple regression model over the plot of BLEU scores against target vocabulary size. This figure shows groups of data points with the same target language in almost vertical lines. Each language pair has a separate parallel training corpus, but the target vocabulary size for one language will be very similar in all of them. The variance in BLEU amongst the group with the same target language is then largely explained by the other factors, similarity and reordering.

Figure 9 shows a simple regression model over the plot of BLEU scores against source vocabulary size.

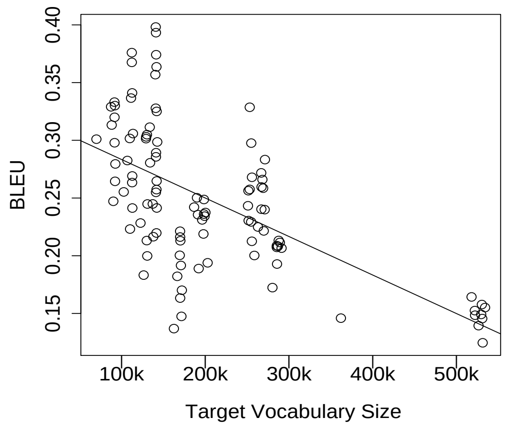


Figure 8. BLEU score of experiments compared to target vocabulary size showing regression

This regression model shows that in isolation source vocabulary size is significant ($p < 0.05$), but that this is due to the distorting effect of Finnish. Excluding results that include Finnish, there is no longer any significant correlation with BLEU. The source morphology might be significant for models trained on smaller data sets, where model parameters are more sensitive to sparse counts.

Figure 10 shows the simple regression model over the plot of BLEU scores against the amount of reordering. This graph shows that with more reordering, the performance of the translation model reduces. Data points with low levels of reordering and high BLEU scores tend to be language pairs where both languages are Romance languages. High BLEU scores with high levels of reordering tend to have German as the source language and a Romance language as the target.

Figure 11 shows the simple regression model over the plot of BLEU scores against the amount of language relatedness. The left hand line of points are the results involving Finnish. The vertical group of points just to the right, are results where Greek is involved. The next set of points are the results where the translation is between Germanic and Romance languages. The final cloud to the right are results where languages are in the same family, either within the Romance or the Germanic languages.

Table 4 shows the amount of the variance of BLEU explained by the different models. As these

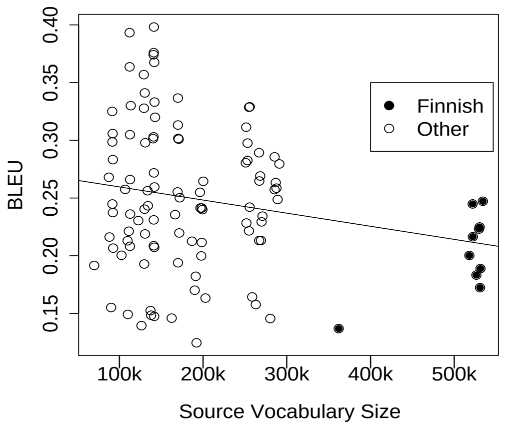


Figure 9. BLEU score of experiments compared to source vocabulary size highlighting the Finnish source vocabulary data points. The regression includes Finnish in the model.

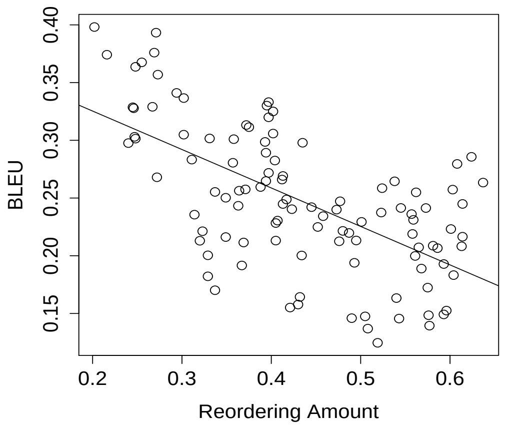


Figure 10. BLEU score of experiments compared to amount of reordering.

Explanatory Variable	R^2	
Target Vocab. Size	**0.388**	***
Reordering Amount	0.384	***
Language Similarity	0.366	***
Source Vocab. Size	0.045	*
Excluding Finnish		
Target Vocab. Size	0.219	***
Reordering Amount	**0.332**	***
Language Similarity	0.188	***
Source Vocab. Size	0.007	

Table 4. Goodness of fit of different simple linear regression models which use just one explanatory variable. The significance level represents the level of probability that the regression is appropriate. The second set of results excludes Finnish in the source and target language.

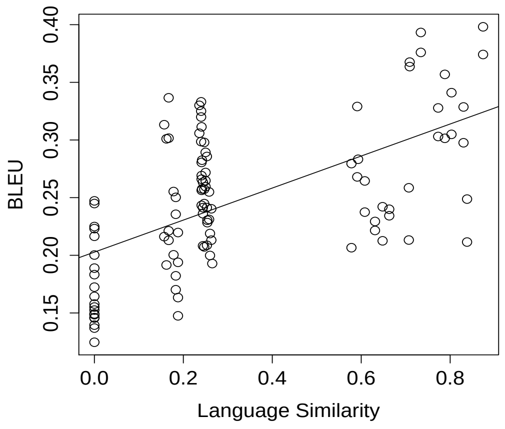


Figure 11. BLEU score of experiments compared to language relatedness.

are simple regression models, with just one explanatory variable, multicolinearity is avoided. This table shows that each of the main effects explains about a third of the variance of BLEU, which means that they can be considered to be of equal importance. When Finnish examples are removed, only reordering retains its power, and target vocabulary and language similarity reduce in importance and source vocabulary size no longer correlates with performance.

8 Conclusion

We have broken down the relative impact of the characteristics of different language pairs on translation performance. The analysis done is able to account for a large percentage (75%) of the variability of the performance of the system, which shows that we have captured the core challenges for the phrase-based model. We have shown that their impact is about the same, with reordering and target vocabulary size each contributing about 0.38%.

These conclusions are only strictly relevant to the model for which this analysis has been performed, the phrase-based model. However, we suspect that the conclusions would be similar for most statistical machine translation models because of their dependence on automatic alignments. This will be the topic of future work.

References

Balthasar Bickel and Johanna Nichols, 2005. *The World Atlas of Language Structures*, chapter Inflectional synthesis of the verb. Oxford University Press.

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

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2008. Further meta-evaluation of machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 70–106, Columbus, Ohio, June. Association for Computational Linguistics.

Isidore Dyen, Joseph Kruskal, and Paul Black. 1992. An indoeuropean classification, a lexicostatistical experiment. *Transactions of the American Philosophical Society*, 82(5).

Chris Dyer. 2007. The 'noisier channel': Translation from morphologically complex languages. In *Proceedings on the Workshop on Statistical Machine Translation*, Prague, Czech Republic.

Sharon Goldwater and David McClosky. 2005. Improving statistical MT through morphological analysis. In *Proceedings of Empirical Methods in Natural Language Processing*.

Philipp Koehn, Franz Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference*, pages 127–133, Edmonton, Canada. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the Association for Computational Linguistics Companion Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of MT-Summit*.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Association for Computational Linguistics*, pages 311–318, Philadelphia, USA.

M Snover, B Dorr, R Schwartz, L Micciulla, and J Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *AMTA*.

Morris Swadesh. 1955. Lexicostatistic dating of prehistoric ethnic contacts. In *Proceedings American Philosophical Society*, volume 96, pages 452–463.

David Talbot and Miles Osborne. 2006. Modelling lexical redundancy for machine translation. In *Proceedings of the Association of Computational Linguistics*, Sydney, Australia.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

An Exploration of Document Impact on Graph-Based Multi-Document Summarization

Xiaojun Wan
Institute of Compute Science and Technology
Peking University
Beijing 100871, China
wanxiaojun@icst.pku.edu.cn

Abstract

The graph-based ranking algorithm has been recently exploited for multi-document summarization by making only use of the sentence-to-sentence relationships in the documents, under the assumption that all the sentences are indistinguishable. However, given a document set to be summarized, different documents are usually not equally important, and moreover, different sentences in a specific document are usually differently important. This paper aims to explore document impact on summarization performance. We propose a document-based graph model to incorporate the document-level information and the sentence-to-document relationship into the graph-based ranking process. Various methods are employed to evaluate the two factors. Experimental results on the DUC2001 and DUC2002 datasets demonstrate that the good effectiveness of the proposed model. Moreover, the results show the robustness of the proposed model.

1 Introduction

Multi-document summarization aims to produce a summary describing the main topic in a document set, without any prior knowledge. Multi-document summary can be used to facilitate users to quickly understand a document cluster. For example, a number of news services (e.g. NewsInEssence[1]) have been developed to group news articles into news topics, and then produce a short summary for each news topic. Users can easily understand the topic they have interest in by taking a look at the short summary, without looking into each individual article within the topic cluster.

[1]http://lada.si.umich.edu:8080/clair/nie1/nie.cgi

Automated multi-document summarization has drawn much attention in recent years. In the communities of natural language processing and information retrieval, a series of workshops and conferences on automatic text summarization (e.g. NTCIR, DUC), special topic sessions in ACL, COLING, and SIGIR have advanced the summarization techniques and produced a couple of experimental online systems.

A particular challenge for multi-document summarization is that a document set might contain diverse information, which is either related or unrelated to the main topic, and hence we need effective summarization methods to analyze the information stored in different documents and extract the globally important information to reflect the main topic. In recent years, both unsupervised and supervised methods have been proposed to analyze the information contained in a document set and extract highly salient sentences into the summary, based on syntactic or statistical features.

Most recently, the graph-based models have been successfully applied for multi-document summarization by making use of the "voting" or "recommendations" between sentences in the documents (Erkan and Radev, 2004; Mihalcea and Tarau, 2005; Wan and Yang, 2006). The model first constructs a directed or undirected graph to reflect the relationships between the sentences and then applies the graph-based ranking algorithm to compute the rank scores for the sentences. The sentences with large rank scores are chosen into the summary. However, the model makes uniform use of the sentences in different documents, i.e. all the sentences are ranked without considering the document-level information and the sentence-to-document relationship. Actually, given a document set, different documents are not equally important. For example, the documents close to the main topics of the document set are usually more important than the documents far away from the main topics of the document set. This document-level information is deemed to have great impact on the sentence ranking process. Moreover, the sentences in the same document cannot be treated uniformly, because some sentences in the document are more important than other sentences because of their different positions in the document or different distances to the document's centroid. In brief, neither the document-level information nor the sentence-to-document relationship has been taken into account in the previous graph-based model.

In order to overcome the limitations of the previous graph-based model, this study proposes the document-based graph model to explore document impact on the graph-based summarization, by incorporating both the document-level information and the sentence-to-document relationship in the graph-based ranking process. We develop various methods to evaluate the document-level information and the sentence-to-document relationship. Experiments on the DUC2001 and DUC2002 datasets have been performed and the results demonstrate the good effectiveness of the proposed model, i.e., the incorporation of document impact can much improve the performance of the graph-based summarization. Moreover, the proposed model is robust with respect to most incorporation schemes.

The rest of this paper is organized as follows. We first introduce the related work in Section 2. The basic graph-based summarization model and the proposed document-based graph model are described in detail in Sections 3 and 4, respectively. We show the experiments and results in Section 5 and finally we conclude this paper in Section 6.

2 Related Work

Generally speaking, summarization methods can be abstractive summarization or extractive summarization. Extractive summarization is a simple but robust method for text summarization and it involves assigning saliency scores to some units (e.g. sentences, paragraphs) of the documents and extracting those with highest scores, while abstraction summarization usually needs information fusion (Barzilay et al., 1999), sentence compression (Knight and Marcu, 2002) and reformulation (McKeown et al., 1999). In this study, we focus on extractive summarization.

The centroid-based method (Radev et al., 2004) is one of the most popular extractive summarization methods. MEAD[2] is an implementation of the centroid-based method that scores sentences based on sentence-level and inter-sentence features, including cluster centroids, position, TFIDF, etc. NeATS (Lin and Hovy, 2002) is a project on multi-document summarization at ISI based on the single-document summarizer-SUMMARIST. Sentence position, term frequency, topic signature and term clustering are used to select important content. MMR (Goldstein et al., 1999) is used to remove redundancy and stigma word filters and time stamps are used to improve cohesion and coherence. To further explore user interface issues, iNeATS (Leuski et al., 2003) is developed based on NeATS. XDoX (Hardy et al., 1998) is a cross document summarizer designed specifically to summarize large document sets. It identifies the most salient themes within the set by passage clustering and then composes an extraction summary, which reflects these main themes. Much other work also explores to find topic themes in the documents for summarization, e.g. Harabagiu and Lacatusu (2005) investigate five different topic representations and introduce a novel representation of topics based on topic themes. In addition, Marcu (2001) selects important sentences based on the discourse structure of the text. TNO's system (Kraaij et al., 2001) scores sentences by combining a unigram language model approach with a Bayesian classifier based on surface features. Nenkova and Louis (2008) investigate how summary length and the characteristics of the input influence the summary quality in multi-document summarization.

Graph-based models have been proposed to rank sentences or passages based on the PageRank algorithm (Page et al., 1998) or its variants. Websumm (Mani and Bloedorn, 2000) uses a graph-connectivity model and operates under the assumption that nodes which are connected to many other nodes are likely to carry salient information. LexPageRank (Erkan and Radev, 2004) is an approach for computing sentence importance based on the concept of eigenvector centrality. It constructs a sentence connectivity matrix and compute sentence importance based on an algorithm similar to PageRank. Mihalcea and Tarau (2005) also propose a similar algorithm based on PageRank to compute sentence importance for document summarization. Wan and Yang (2006) improve the ranking algo-

[2] http://www.summarization.com/mead/

rithm by differentiating intra-document links and inter-document links between sentences. All these methods make use of the relationships between sentences and select sentences according to the "votes" or "recommendations" from their neighboring sentences, which is similar to PageRank.

Other related work includes topic-focused multi-document summarization (Daumé. and Marcu, 2006; Gupta et al., 2007; Wan et al., 2007), which aims to produce summary biased to a given topic or query. It is noteworthy that our proposed approach is inspired by (Liu and Ma, 2005), which proposes the Conditional Markov Random Walk Model based on two-layer web graph in the tasks of web page retrieval.

3 The Basic Graph-Based Model (GM)

The basic graph-based model is essentially a way of deciding the importance of a vertex within a graph based on global information recursively drawn from the entire graph. The basic idea is that of "voting" or "recommendation" between the vertices. A link between two vertices is considered as a vote cast from one vertex to the other vertex. The score associated with a vertex is determined by the votes that are cast for it, and the score of the vertices casting these votes.

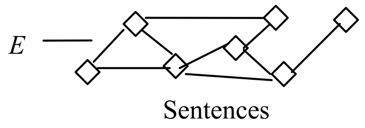


Figure 1. One-layer link graph

Formally, given a document set D, let G=(V, E) be an undirected graph to reflect the relationships between sentences in the document set, as shown in Figure 1. V is the set of vertices and each vertex v_i in V is a sentence in the document set. E is the set of edges. Each edge e_{ij} in E is associated with an affinity weight $f(v_i, v_j)$ between sentences v_i and v_j ($i{\neq}j$). The weight is computed using the standard cosine measure between the two sentences.

$$f(v_i, v_j) = sim_{\cos ine}(v_i, v_j) = \frac{\vec{v}_i \cdot \vec{v}_j}{|\vec{v}_i| \times |\vec{v}_j|} \tag{1}$$

where $\vec{v}_i$ and $\vec{v}_j$ are the corresponding term vectors of v_i and v_j. Here, we have $f(v_i, v_j)$=f(v_j, v_i). Two vertices are connected if their affinity weight is larger than 0 and we let $f(v_i, v_i)$=0 to avoid self transition.

We use an affinity matrix M to describe G with each entry corresponding to the weight of an edge in the graph. $M = (M_{i,j})_{|V|\times|V|}$ is defined as follows:

$$M_{i,j} = \begin{cases} f(v_i, v_j), & \text{if } v_i \text{ and } v_j \text{ is connected} \\ & \text{and } \; i \neq j; \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Then M is normalized to $\widetilde{M}$ as follows to make the sum of each row equal to 1:

$$\widetilde{M}_{i,j} = \begin{cases} M_{i,j} \Big/ \sum_{j=1}^{|V|} M_{i,j}, & \text{if } \sum_{j=1}^{|V|} M_{i,j} \neq 0 \\ 0 & , \text{ otherwise} \end{cases} \tag{3}$$

Based on matrix $\widetilde{M}$, the saliency score *SenScore*(v_i) for sentence v_i can be deduced from those of all other sentences linked with it and it can be formulated in a recursive form as in the PageRank algorithm:

$$SenScore(v_i) = \mu \cdot \sum_{all\, j \neq i} SenScore(v_j) \cdot \widetilde{M}_{j,i} + \frac{(1-\mu)}{|V|} \tag{4}$$

And the matrix form is:

$$\vec{\lambda} = \mu \widetilde{M}^T \vec{\lambda} + \frac{(1-\mu)}{|V|} \vec{e} \tag{5}$$

where $\vec{\lambda} = [SenScore(v_i)]_{|V|\times 1}$ is the vector of sentence saliency scores. $\vec{e}$ is a vector with all elements equaling to 1. μ is the damping factor usually set to 0.85, as in the PageRank algorithm.

The above process can be considered as a Markov chain by taking the sentences as the states and the corresponding transition matrix is given by $A = \mu \widetilde{M}^T + \frac{(1-\mu)}{|V|} \vec{e}\vec{e}^T$. The stationary probability distribution of each state is obtained by the principal eigenvector of the transition matrix.

For implementation, the initial scores of all sentences are set to 1 and the iteration algorithm in Equation (4) is adopted to compute the new scores of the sentences. Usually the convergence of the iteration algorithm is achieved when the difference between the scores computed at two successive iterations for any sentences falls below a given threshold (0.0001 in this study).

We can see that the basic graph-based model is built on the single-layer sentence graph and the transition probability between two sentences in the Markov chain depends only on the sentences themselves, not taking into account the document-level information and the sentence-to-document relationship.

4 The Document-Based Graph Model (DGM)

4.1 Overview

As we mentioned in previous section, there may be many factors that can have impact on the importance analysis of the sentences. This study aims to examine the document impact by incorporating the document importance and the sentence-to-document correlation into the sentence ranking process. Our assumption is that the sentences, which belong to an important document and are highly correlated with the document, will be more likely to be chosen into the summary.

In order to incorporate the document-level information and the sentence-to-document relationship, the document-based graph model is proposed based on the two-layer link graph including both sentences and documents. The novel representation is shown in Figure 2. As can be seen, the lower layer is just the traditional link graph between sentences that has been well studied in previous work. And the upper layer represents the documents. The dashed lines between these two layers indicate the conditional influence between the sentences and the documents.

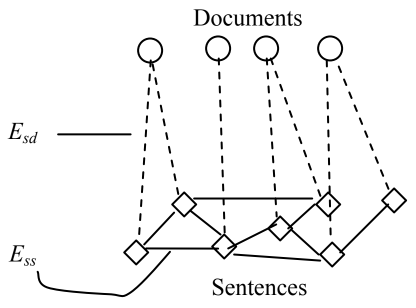


Figure 2. Two-layer link graph

Formally, the new representation for the two-layer graph is denoted as $G^*=<V_s, V_d, E_{ss}, E_{sd}>$, where $V_s=V=\{v_i\}$ is the set of sentences and $V_d=D=\{d_j\}$ is the set of documents; $E_{ss}=E=\{e_{ij}|v_i, v_j\in V_s\}$ includes all possible links between sentences and $E_{sd}=\{e_{ij}|v_i\in V_s, d_j\in V_d$ and $d_j=doc(v_i)\}$ includes the correlation link between any sentence and its belonging document. Here, we use $doc(v_i)$ to denote the document containing sentence v_i. For further discussions, we let $\pi(doc(v_i)) \in[0,1]$ denote the importance of document $doc(v_i)$ in the document set, and let $\omega(v_i, doc(v_i)) \in[0,1]$ denote the strength of the correlation between sentence v_i and its document $doc(v_i)$.

The two factors are incorporated into the affinity weight between sentences and the new sentence-to-sentence affinity weight is denoted as $f(v_i, v_j|doc(v_i), doc(v_j))$, which is conditioned on the two documents containing the two sentences. The new conditional affinity weight is computed by linearly combining the affinity weight conditioned on the first document (i.e. $f(v_i,v_j|doc(v_i))$) and the affinity weight conditioned on the second document (i.e. $f(v_i,v_j|doc(v_j))$).

Formally, the conditional affinity weight is computed as follows to incorporate the two factors:

$$\begin{aligned}
& f(v_i,v_j \mid doc(v_i),doc(v_j)) \\
&= \lambda\cdot f(v_i,v_j \mid doc(v_i)) + (1-\lambda)\cdot f(v_i,v_j \mid doc(v_j)) \\
&= \lambda\cdot f(v_i,v_j)\cdot\pi(doc(v_i))\cdot\omega(v_i,doc(v_i)) \\
&\quad + (1-\lambda)\cdot f(v_i,v_j)\cdot\pi(doc(v_j))\cdot\omega(v_j,doc(v_j)) \\
&= f(v_i,v_j)\cdot(\lambda\cdot\pi(doc(v_i))\cdot\omega(v_i,doc(v_i)) \\
&\quad + (1-\lambda)\cdot\pi(doc(v_j))\cdot\omega(v_j,doc(v_j))) \\
&= sim_{\cos ine}(v_i,v_j)\cdot(\lambda\cdot\pi(doc(v_i))\cdot\omega(v_i,doc(v_i)) \\
&\quad + (1-\lambda)\cdot\pi(doc(v_j))\cdot\omega(v_j,doc(v_j)))
\end{aligned} \tag{6}$$

where $\lambda\in[0,1]$ is the combination weight controlling the relative contributions from the first document and the second document. Note that usually $f(v_i, v_j|doc(v_i), doc(v_j))$ is not equal to $f(v_j, v_i|doc(v_j), doc(v_i))$, but the two scores are equal when λ is set to 0.5. Various methods can be used to evaluate the document importance and the sentence-document correlation, which will be described in next sections.

The new affinity matrix M^* is then constructed based on the above conditional sentence-to-sentence affinity weight.

$$M^*_{i,j} = \begin{cases} f(v_i,v_j \mid doc(v_i),doc(v_j)), & \text{if if } v_i \text{ and } v_j \text{ is connected} \\ & \text{and } i\neq j \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

Likewise, M^* is normalized to $\tilde{M}^*$ and the iterative computation as in Equation (4) is then based on $\tilde{M}^*$. The transition matrix in the Markov chain is then denoted by $A^* = \mu\tilde{M}^{*T} + \frac{(1-\mu)}{|V|}\vec{e}\vec{e}^{\,T}$ and the sentence scores is obtained by the principle eigenvector of the new transition matrix A^*.

4.2 Evaluating Document Importance (π)

The function $\pi(doc(v_i))$ aims to evaluate the importance of document $doc(v_i)$ in the document set D. The following three methods are developed to evaluate the document importance.

***π_1*:** It uses the cosine similarity value between the document and the whole document set as the importance score of the document[3]:

$$\pi_1(doc(v_i)) = sim_{\cos ine}(doc(v_i), D) \quad (8)$$

***π_2*:** It uses the average similarity value between the document and any other document in the document set as the importance score of the document:

$$\pi_2(doc(v_i)) = \frac{\sum_{d' \in D \text{ and } d' \neq doc(v_i)} sim_{\cos ine}(doc(v_i), d')}{|D| - 1} \quad (9)$$

***π_3*:** It constructs a weighted graph between documents and uses the PageRank algorithm to compute the rank scores of the documents as the importance scores of the documents. The link weight between two documents is computed using the cosine measure. The equation for iterative computation is the same with Equation (4).

4.3 Evaluating Sentence-Document Correlation (*ω*)

The function *ω*(v_i, *doc*(v_i)) aims to evaluate the correlation between sentence v_i and its document *doc*(v_i). The following four methods are developed to compute the strength of the correlation. The first three methods are based on sentence position in the document, under the assumption that the first sentences in a document are usually more important than other sentences. The last method is based on the content similarity between the sentence and the document.

***ω_1*:** The correlation strength between sentence v_i and its document *doc*(v_i) is based on the position of the sentence as follows:

$$\omega_1(v_i, doc(v_i)) = \begin{cases} 1 \text{ if } pos(v_i) \leq 3 \\ 0.5 \text{ Otherwise} \end{cases} \quad (10)$$

where *pos*(v_i) returns the position number of sentence v_i in its document. For example, if v_i is the first sentence in its document, *pos*(v_i) is 1.

***ω_2*:** The correlation strength between sentence v_i and its document *doc*(v_i) is based on the position of the sentence as follows:

$$\omega_2(v_i, doc(v_i)) = 1 - \frac{pos(v_i) - 1}{sen\_count(doc(v_i))} \quad (11)$$

where *sen_count*(*doc*(v_i)) returns the total number of sentences in document *doc*(v_i).

***ω_3*:** The correlation strength between sentence v_i and its document *doc*(v_i) is based on the position of the sentence as follows:

$$\omega_3(v_i, doc(v_i)) = 0.5 + \frac{1}{pos(v_i) + 1} \quad (12)$$

***ω_4*:** The correlation strength between sentence v_i and its document *doc*(v_i) is based on the cosine similarity between the sentence and the document:

$$\omega_4(v_i, doc(v_i)) = sim_{\cos ine}(v_i, doc(v_i)) \quad (13)$$

5 Empirical Evaluation

5.1 Dataset and Evaluation Metric

Generic multi-document summarization has been one of the fundamental tasks in DUC 2001[4] and DUC 2002[5] (i.e. task 2 in DUC 2001 and task 2 in DUC 2002), and we used the two tasks for evaluation. DUC2001 provided 30 document sets and DUC 2002 provided 59 document sets (D088 is excluded from the original 60 document sets by NIST) and generic abstracts of each document set with lengths of approximately 100 words or less were required to be created. The documents were news articles collected from TREC-9. The sentences in each article have been separated and the sentence information has been stored into files. The summary of the two datasets are shown in Table 1.

	DUC 2001	**DUC 2002**
Task	Task 2	Task 2
Number of documents	309	567
Number of clusters	30	59
Data source	TREC-9	TREC-9
Summary length	100 words	100 words

Table 1. Summary of datasets

We used the ROUGE (Lin and Hovy, 2003) toolkit (i.e. ROUGEeval-1.4.2 in this study) for evaluation, which has been widely adopted by DUC for automatic summarization evaluation. It measured summary quality by counting overlapping units such as the n-gram, word sequences and word pairs between the candidate summary and the reference summary. ROUGE-N was an n-gram recall measure computed as follows:

$$ROUGE-N = \frac{\sum_{S \in \{\text{Ref Sum}\}} \sum_{n\text{-}gram \in S} Count_{match}(n-gram)}{\sum_{S \in \{\text{Ref Sum}\}} \sum_{n\text{-}gram \in S} Count(n-gram)} \quad (14)$$

[3] A document set is treated as a single text by concatenating all the document texts in the set.

[4] http://www-nlpir.nist.gov/projects/duc/guidelines/2001.html
[5] http://www-nlpir.nist.gov/projects/duc/guidelines/2002.html

where n stood for the length of the n-gram, and $Count_{match}(n\text{-}gram)$ was the maximum number of n-grams co-occurring in a candidate summary and a set of reference summaries. *Count*(*n-gram*) was the number of n-grams in the reference summaries.

ROUGE toolkit reported separate scores for 1, 2, 3 and 4-gram, and also for longest common subsequence co-occurrences. Among these different scores, unigram-based ROUGE score (ROUGE-1) has been shown to agree with human judgment most (Lin and Hovy. 2003). We showed three of the ROUGE metrics in the experimental results: ROUGE-1 (unigram-based), ROUGE-2 (bigram-based), and ROUGE-W (based on weighted longest common subsequence, weight=1.2). In order to truncate summaries longer than length limit, we used the "-l" option in ROUGE toolkit. We also used the "-m" option for word stemming.

5.2 Evaluation Results

In the experiments, the combination weight λ for the proposed summarization model is typically set to 0.5 without tuning, i.e. the two documents for two sentences have equal influence on the summarization process. Note that after the saliency scores of sentences have been obtained, a greedy algorithm (Wan and Yang, 2006) is applied to remove redundancy and finally choose both informative and novel sentences into the summary. The algorithm is actually a variant version of the MMR algorithm (Goldstein et al., 1999).

The proposed document-based graph model (denoted as DGM) with different settings is compared with the basic graph-based Model (denoted as GM), the top three performing systems and two baseline systems on DUC2001 and DUC2002, respectively. The top three systems are the systems with highest ROUGE scores, chosen from the performing systems on each task respectively. The lead baseline and coverage baseline are two baselines employed in the generic multi-document summarization tasks of DUC2001 and DUC2002. The lead baseline takes the first sentences one by one in the last document in the collection, where documents are assumed to be ordered chronologically. And the coverage baseline takes the first sentence one by one from the first document to the last document. Tables 2 and 3 show the comparison results on DUC2001 and DUC2002, respectively. In Table 1, SystemN, SystemP and System T are the top three performing systems for DUC2001. In Table 2, System19, System26, System28 are the top three performing systems for DUC2002. The document-based graph model is configured with different settings (i.e. π_1-π_3, ω_1-ω_4). For example, DGM($\pi_1+\omega_1$) refers to the DGM model with π_1 to evaluate the document importance and ω_1 to evaluate the correlation between a sentence and its document.

System	ROUGE-1	ROUGE-2	ROUGE-W
DGM($\pi_1+\omega_1$)	0.35658	0.05926	0.10712
DGM($\pi_1+\omega_2$)	0.35945	0.06304*	0.10820
DGM($\pi_1+\omega_3$)	0.36349*	0.06472*	0.10952
DGM($\pi_1+\omega_4$)	0.35421	0.05934	0.10695
DGM($\pi_2+\omega_1$)	0.35555	0.06554*	0.10924
DGM($\pi_2+\omega_2$)	0.37228*	0.06787*	0.11295*
DGM($\pi_2+\omega_3$)	0.37347*	0.06612*	0.11352*
DGM($\pi_2+\omega_4$)	0.36340	0.06397*	0.11006
DGM($\pi_3+\omega_1$)	0.35333	0.06353*	0.10834
DGM($\pi_3+\omega_2$)	0.37082*	0.06708*	0.11235
DGM($\pi_3+\omega_3$)	0.37056*	0.06503*	0.11227*
DGM($\pi_3+\omega_4$)	0.36667*	0.06585*	0.11114
GM	0.35527	0.05608	0.10641
SystemN	0.33910	0.06853	0.10240
SystemP	0.33332	0.06651	0.10068
SystemT	0.33029	0.07862	0.10215
Coverage	0.33130	0.06898	0.10182
Lead	0.29419	0.04033	0.08880

Table 2. Comparison results on DUC2001

System	ROUGE-1	ROUGE-2	ROUGE-W
DGM($\pi_1+\omega_1$)	0.37891	0.08398	0.12390
DGM($\pi_1+\omega_2$)	0.39013*	0.08770*	0.12726*
DGM($\pi_1+\omega_3$)	0.38490*	0.08355	0.12570
DGM($\pi_1+\omega_4$)	0.38464	0.08371	0.12443
DGM($\pi_2+\omega_1$)	0.38296	0.08369	0.12499
DGM($\pi_2+\omega_2$)	0.38143	0.08792*	0.12506
DGM($\pi_2+\omega_3$)	0.38177	0.08624*	0.12511
DGM($\pi_2+\omega_4$)	0.38576*	0.08167	0.12611
DGM($\pi_3+\omega_1$)	0.38079	0.08391	0.12392
DGM($\pi_3+\omega_2$)	0.38103	0.08608*	0.12446
DGM($\pi_3+\omega_3$)	0.38236	0.08675*	0.12478
DGM($\pi_3+\omega_4$)	0.38719*	0.08150	0.12633*
GM	0.37595	0.08304	0.12173
System26	0.35151	0.07642	0.11448
System19	0.34504	0.07936	0.11332
System28	0.34355	0.07521	0.10956
Coverage	0.32894	0.07148	0.10847
Lead	0.28684	0.05283	0.09525

Table 3. Comparison results on DUC2002
(* indicates that the improvement over the baseline GM model is statistically significant at 95% confidence level)

Seen from the tables, the proposed document-based graph model with different settings can outperform the basic graph-based model and other baselines over almost all three metrics on both

DUC2001 and DUC2002 datasets. The results demonstrate the good effectiveness of the proposed model, i.e. the incorporation of document impact does benefit the graph-based summarization model. It is interesting that the three methods for computing document importance and the four methods for computing the sentence-document correlation are almost as effective as each other on the DUC2002 dataset. However, π_1 does not perform as well as π_2 and π_3, and ω_1 and ω_4 does not perform as well as ω_2 and ω_3 on the DUC2001 dataset.

In order to investigate the relative contributions from the two documents for two sentences to the summarization performance, we varies the combination weight λ from 0 to 1 and Figures 3-6 show the ROUGE-1 and ROUGE-W curves on DUC2001 and DUC2002 respectively. The similar ROUGE-2 curves are omitted here.

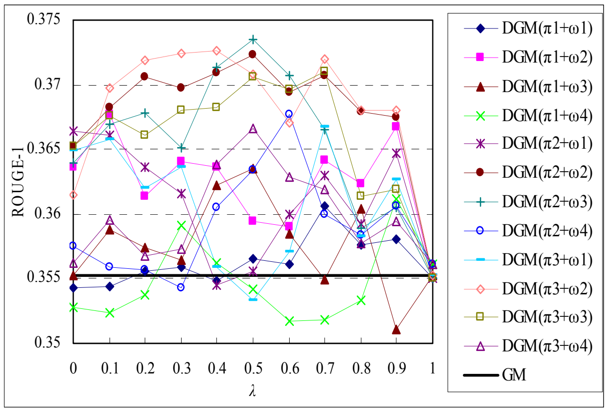


Figure 3. ROUGE-1 vs. λ on DUC2001

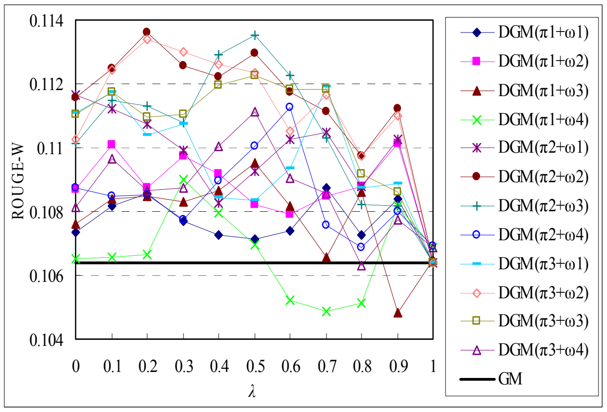


Figure 4. ROUGE-W vs. λ on DUC2001

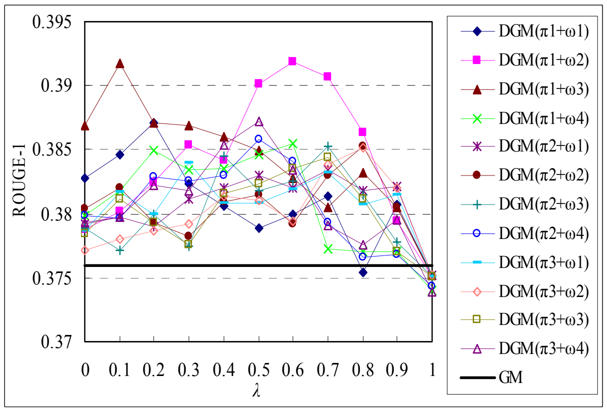


Figure 5. ROUGE-1 vs. λ on DUC2002

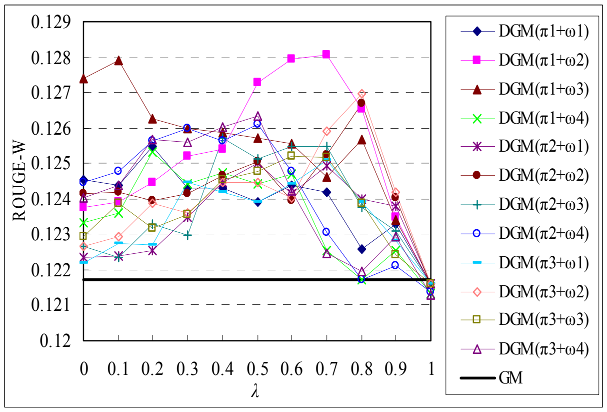


Figure 6. ROUGE-W vs. λ on DUC2002

We can see from the figures that the proposed document-based graph model with different settings can almost always outperform the basic graph-based model, with respect to different values of λ. The results show the robustness of the proposed model. We can also see that for most settings of the propose model, very large values or very small values of λ can deteriorate the summarization performance, i.e. both the first document and the second document in the computation of the conditional affinity weight between sentences have great impact on the summarization performance.

6 Conclusion and Future Work

This paper examines the document impact on the graph-based model for multi-document summarization. The document-level information and the sentence-to-document relationship are incorporated into the graph-based ranking algorithm. The experimental results on DUC2001 and DUC2002 demonstrate the good effectiveness of the proposed model.

In this study, we directly make use of the coarse-grained document-level information. Actually, a document can be segmented into a few subtopic passages by using the TextTiling algorithm (Hearst, 1997), and we believe the subtopic passage is more fine-grained than the original document. In future work, we will exploit this kind of subtopic-level information to further improve the summarization performance.

Acknowledgments

This work was supported by the National Science Foundation of China (No.60703064), the Research Fund for the Doctoral Program of Higher Education of China (No.20070001059) and the National High Technology Research and Development Program of China (No.2008AA01Z421). We also thank the anonymous reviewers for their useful comments.

References

R. Barzilay, K. R. McKeown and M. Elhadad. 1999. Information fusion in the context of multi-document summarization. In Proceedings of ACL1999.

H. Daumé and D. Marcu. 2006. Bayesian query-focused summarization. 2006. In Proceedings of COLING-ACL2006.

G. Erkan and D. Radev. 2004. LexPageRank: prestige in multi-document text summarization. In Proceedings of EMNLP'04.

J. Goldstein, M. Kantrowitz, V. Mittal and J. Carbonell. 1999. Summarizing text documents: sentence selection and evaluation metrics. In Proceedings of SIGIR-99.

S. Gupta, A. Nenkova and D. Jurafsky. 2007. Measuring importance and query relevance in topic-focused multi-document summarization. In Proceedings of ACL-07.

S. Harabagiu and F. Lacatusu. 2005. Topic themes for multi-document summarization. In Proceedings of SIGIR'05.

H. Hardy, N. Shimizu, T. Strzalkowski, L. Ting, G. B. Wise. and X. Zhang. 2002. Cross-document summarization by concept classification. In Proceedings of SIGIR'02.

M. Hearst. 1997. TextTiling: segmenting text into multi-paragraph subtopic passages. Computational Linguistics, 23(1): 33-64.

K. Knight. and D. Marcu. 2002. Summarization beyond sentence extraction: a probabilistic approach to sentence compression, Artificial Intelligence, 139(1).

W. Kraaij, M. Spitters and M. van der Heijden. 2001. Combining a mixture language model and Naïve Bayes for multi-document summarization. In SIGIR 2001 Workshop on Text Summarization.

A. Leuski, C.-Y. Lin and E. Hovy. 2003. iNeATS: interactive multi-document summarization. In Proceedings of ACL2003.

C.-Y. Lin and E. H. Hovy. 2002. From single to multi-document summarization: a prototype system and its evaluation. In Proceedings of ACL-2002.

C.-Y. Lin and E. H. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In Proceedings of HLT-NAACL2003.

T.-Y. Liu and W.-Y. Ma. 2005. Webpage importance analysis using Conditional Markov Random Walk. In Proceedings of WI2005.

I. Mani and E. Bloedorn. 2000. Summarizing similarities and differences among related documents. Information Retrieval, 1(1).

D. Marcu. Discourse-based summarization in DUC–2001. 2001. In SIGIR 2001 Workshop on Text Summarization.

K. McKeown, J. Klavans, V. Hatzivassiloglou, R. Barzilay and E. Eskin. 1999. Towards multidocument summarization by reformulation: progress and prospects, in Proceedings of AAAI1999.

R. Mihalcea and P. Tarau. 2005. A language independent algorithm for single and multiple document summarization. In Proceedings of IJCNLP'2005.

A. Nenkova and A. Louis. 2008. Can you summarize this? Identifying correlates of input difficulty for generic multi-document summarization. In Proceedings of ACL-08: HLT.

L. Page, S. Brin, R. Motwani and T. Winograd. 1998. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Libraries.

D. R. Radev, H. Y. Jing, M. Stys and D. Tam. 2004. Centroid-based summarization of multiple documents. Information Processing and Management, 40: 919-938.

X. Wan and J. Yang. 2006. Improved affinity graph based multi-document summarization. In Proceedings of HLT-NAACL2006.

X. Wan, J. Yang and J. Xiao. 2007. Manifold-ranking based topic-focused multi-document summarization. In Proceedings of IJCAI2007.

Topic-Driven Multi-Document Summarization with Encyclopedic Knowledge and Spreading Activation

Vivi Nastase
EML Research gGmbH
Heidelberg, Germany
nastase@eml-research.de

Abstract

Information of interest to users is often distributed over a set of documents. Users can specify their request for information as a query/topic – a set of one or more sentences or questions. Producing a good summary of the relevant information relies on understanding the query and linking it with the associated set of documents. To "understand" the query we expand it using encyclopedic knowledge in Wikipedia. The expanded query is linked with its associated documents through spreading activation in a graph that represents words and their grammatical connections in these documents. The topic expanded words and activated nodes in the graph are used to produce an extractive summary. The method proposed is tested on the DUC summarization data. The system implemented ranks high compared to the participating systems in the DUC competitions, confirming our hypothesis that encyclopedic knowledge is a useful addition to a summarization system.

1 Introduction

Topic-driven summarization reflects a user-based summarization task: from a set of documents derive a summary that contains information on a specific topic of interest to a user. Producing a good summary relies on "understanding" the user's information request, and the documents to be summarized. It is commonly agreed that the verbal part of a text provides pointers to a much larger body of knowledge we assume the listener has. An American citizen, for example, when told *There will be fireworks on July 4^{th}*, understands that there will be a celebration involving fireworks on the occasion of the U.S. Independence Day. Understanding an utterance implies lexical, common-sense and encyclopedic knowledge. Lexical knowledge is usually incorporated in systems through machine readable dictionaries, wordnets or thesauri. Common-sense and encyclopedic knowledge were harder to capture, but recently Wikipedia has opened the possibility of accessing such knowledge on a large scale, and in numerous languages.

To "understand" a user's information request – one or more sentences or questions (the *topic* of the summary) – summarization systems try to expand it. This will provide later stages of processing with more keywords/keyphrases for retrieving from the documents relevant fragments. In this paper we experiment with Wikipedia for topic expansion. The body of research involving Wikipedia as a source of knowledge is growing fast, as the NLP community finds more and more applications of this useful resource: it is used to acquire knowledge (Suchanek et al., 2007; Auer et al., 2007); to induce taxonomies and compute semantic relatedness (Ponzetto & Strube, 2007b; 2007a); as a source of features for text classification (Gabrilovich & Markovitch, 2006) and for answering questions (Ahn et al., 2004; Katz et al., 2005). The work presented here uses hyperlinks in Wikipedia articles to expand keywords and keyphrases extracted from the query. Ambiguous words are disambiguated using the context provided by the query.

"Understanding" the documents to be summarized implies identifying the entities mentioned, how they are connected, and how they are related to the entities in the topic. For this, we start again from the topic, and spread an activation signal in a large graph that covers all documents for this topic – nodes are words/named entities in the texts, links are grammatical relations. This way we cross from the topic to the documents, and combine information which is important in the topic with information which is important and relevant in the documents. We take the most highly activated nodes as additional topic expansions, and produce an extractive summary by choosing from the sentences that connect the topic expansion words in the large document graph.

The experiments confirm that Wikipedia is a source of useful knowledge for summarization, and that further expanding the topic within the associated set of documents improves the summarization results even more. We compare the performance of the summarization system to that of participating systems in the DUC competitions. The system we describe ranks 2^{nd}, 9^{th} and 5^{th} in terms of ROUGE-SU4 on the DUC 2005, DUC 2006 and DUC 2007 data respectively.

2 Related Work

While the recent exponential increase in the amount of information with which we must cope makes summarization a very desirable tool in the present, summarization is not a novel task. Rath et al. (1961) and Edmundson (1969) have explored extractive summary formation, and have raised important evaluation issues for extractive summaries when compared to several human produced gold standards. Nowadays, summarization methods try to incorporate tools, methodologies and resources developed over the past decades. The NIST organized competitions under the Document Understanding Conferences – DUC (since 2008, Text Analysis Conference (TAC))[1] events provide a forum for the comparison of a variety of approaches, ranging from knowledge poor – Gotti et al. (2007) rely exclusively on a parser, without any additional sources of information – to knowledge rich and complex – GISTexter (Hickl et al., 2007) combines question answering, textual entailment, topic signature modules and a variety of knowledge sources for summarization.

[1]`http://duc.nist.gov/`, `http://www.nist.gov/tac`.

The most frequently used knowledge source in NLP in general, and also for summarization, is WordNet (Fellbaum, 1998). Barzilay & Elhadad (1999) use WordNet to model a text's content relative to a topic based on lexical chains. The sentences intersected by the most and strongest chains are chosen for the extractive summary. Alternative sources for query expansion and document processing have also been explored. Amini & Usunier (2007) use the documents to be summarized themselves to cluster terms, and thus expanding the query "internally". More advanced methods for query expansion use "topic signatures" – words and grammatically related pairs of words that model the query and even the expected answer from sets of documents marked as relevant or not (Lin & Hovy, 2000; Harabagiu, 2004).

Graph-based methods for text summarization work usually at the level of sentences (Erkan & Radev, 2004; Mihalcea & Tarau, 2004). Edge weights between sentences represent a similarity measure, and a PageRank algorithm is used to determine the sentences that are the most salient from a collection of documents and closest to a given topic. At the word level, Leskovec et al. (2004) build a document graph using subject-verb-object triples, semantic normalization and coreference resolution. They use several methods (node degree, PageRank, Hubs, etc.) to compute statistics for the nodes in the network, and use these as attribute values in a machine learning algorithm, where the attribute that is learned is whether the node should appear in the final summary or not. Annotations for training come from human produced summaries. Mohamed & Rajasekaran (2006) incrementally build a graph for a document collection by combining graph-representations of sentences. Links between entities in a sentence can be *isa* (within an NP) or *related_to* (between different phrases in a sentence). Nodes and relations are weighted according to their connectivity, and sentence selection for the final summary is based on the most highly connected nodes. Ye & Chua (2006) build an extractive summary based on a concept lattice, which captures in a hierarchical structure co-occurrences of concepts among sentences. Nodes higher in this structure correspond to frequently co-occurring terms, and are

```
<topic>
<num> D0704A < /num>
<title> Amnesty International < /title>
<narr>
What is the scope of operations of Amnesty International
and what are the international reactions to its activities?
Give examples of charges lodged by the organization and
complaints against it.
< /narr>
<docs>
...
< /docs>
< /topic>
```

```
<topic>
<num> D0740I < /num>
<title> round-the-world balloon flight < /title>
<narr>
Report on the planning, attempts and first success-
ful balloon circumnavigation of the earth by Bertrand
Piccard and his crew.
< /narr>
<docs>
...
< /docs>
< /topic>
```

Figure 1: Sample topics from DUC 2007

assumed to be more representative with respect to the document topic.

Mani & Bloedorn (1999) build a "chronological" graph, in which sentence order is respected and each occurrence of a concept is a separate node. Edges between nodes cover several types of relations: adjacency (ADJ); identity – instance of the same word (SAME); other semantic links, in particular synonymy and hypernymy; PHRASE links connect components of a phrase; NAME indicate named entities; COREF link coreferential name instances. Among other things, they identify regions of the text salient to a user's query, based on spreading activation starting from query words in this document graph. Spreading activation was introduced in the 60s and 70s to model psychological processes of memory activation in humans (Quillian, 1967; Collins & Loftus, 1975).

In this approach we use Wikipedia as a source of knowledge for related concepts – the texts of hyperlinks in an article describing a concept are taken as its related concepts. The query is further expanded by using spreading activation to move away from the topic in a large graph that covers all documents for a given topic. From the nodes thus reached we select using a PageRank algorithm the ones that are most important in the documents. We study the impact of a decay parameter which controls how far to move from the topic, and the number of highest ranked nodes to be added to the expanded topic. The summary is built based on word associations in the documents' graph.

3 Topic Expansion with Encyclopedic Knowledge or WordNet

In DUC topic-driven multi-document summarization, the topic has a title, an ID that links it to a set of documents, and one or more sentences and/or questions, as illustrated in Figure 1.

Topic processing is done in several steps:

1. Preprocessing: Produce the dependency pair representation of the topics using the Stanford Parser[2]. Pairs that have closed-class words are filtered out, and the remaining words are lemmatized[3]. We extract named entities (NEs), as the parser works at the word level. In the dependency pairs we replace an NE's fragments with the complete NE.

2a. Query expansion with Wikipedia: Extract all open-class words and NEs from the topic, and expand them using Wikipedia articles whose titles are these words or phrases.

For each Wikipedia article we extract as related concepts the texts of the hyperlinks in the first paragraph (see Figure 2[4]). The reason for not including links from the entire article body is that apart from the first paragraph, which is more focused, often times hyperlinks are included whenever the underlying concept appears in Wikipedia, without it being

[2]`http://nlp.stanford.edu/software/lex-parser.shtml`

[3]Using XTAG morphological database `ftp://ftp.cis.upenn.edu/pub/xtag/morph-1.5/morph-1.5.tar.gz`.

[4]The left side shows the first paragraph as it appears on the page, the right side shows the corresponding fragment from the source file, with the annotations specific to Wikipedia.

Mining

Mining is the extraction of valuable minerals or other geological materials from the earth, usually (but not always) from an ore body, vein or (coal) seam. Materials recovered by mining include bauxite, coal, copper, gold, silver, diamonds, iron, precious metals, lead, limestone, magnesite, nickel, phosphate, oil shale, rock salt, tin, uranium and molybdenum . Any material that cannot be grown from agricultural processes, or created artificially in a laboratory or factory , is usually mined. Mining in a wider sense comprises extraction of any non-renewable resource (e.g. petroleum, natural gas ,or even water).

```
'''Mining''' is the extraction of [[value
(economics)|valuable]] [[mineral]]s or other
[[geology|geological]] materials from the
earth, usually (but not always) from an
[[ore]] body, [[vein (geology)|vein]] or (coal)
seam.  Materials recovered by mining include
[[bauxite]], [[coal]], [[copper]], [[gold]],
[[silver]], [[diamond]]s, [[iron]], [[precious
metal]]s, [[lead]], [[limestone]], [[magnesite]],
[[nickel]], [[phosphate]], [[oil shale]], [[Sodium
chloride|rock salt]], [[tin]], [[uranium]] and
[[molybdenum]].  Any material that cannot be grown
from [[agriculture|agricultural]] processes, or
created [[Chemical synthesis|artificially]] in a
[[laboratory]] or [[factory]], is usually mined.
Mining in a wider sense comprises extraction of any
[[non-renewable resource]] (e.g., [[petroleum]],
[[natural gas]], or even [[fossil water|water]]).
```

Extracted related concepts for *mining*:

value (economics), valuable, mineral, geology, geological, ore, vein (geology), vein, coal, bauxite, copper, gold, silver, diamond, iron, precious metal, lead, limestone, magnesite, nickel, phosphate, oil shale, Sodium chloride, rock salt, agriculture, agricultural, Chemical synthesis, artificially, laboratory, factory, non-renewable resource, petroleum, natural gas, fossil water, water.

Figure 2: First paragraph for article *Mining* in the English Wikipedia, and the extracted related concepts.

Word	Wikipedia expansion	WordNet expansion
mining	lead, agricultural, mineral, gold, ore, petroleum, nickel, iron, coal, tin, value, copper, water, bauxite, silver, diamond	production
flight	lift, air	pass, trip, lam, overflight, ballooning, nonstop flight, aviation, soaring, air, flying, solo, break, escape
status	registered	way, situation, mode, position, place, par, need, light, danger, health, state, standing, face, rank, demand, command, control
Southern Poverty Law Center	racism, American, United States, research, civil rights, litigation	–

Table 1: Expanded concepts from DUC 2007 topics, after filtering based on the documents to be summarized.

particularly relevant to the current article.

To expand a word (or NE) W from the query, we search for an article having W as the title, or part of the title.

1. If one exact match is found (e.g. Southern Poverty Law Center), extract the related concepts for this article.

2. If several exact or partial matches are found, use the larger context of the query to narrow down to the intended meaning. For example, *Turkey* – referring to the country – appears in several topics in the DUC 2007 data. There are multiple entries for "Turkey" in Wikipedia – for the country, the bird, cities with this name in the U.S. among others. We use a Lesk-like measure, and compute the overlap between the topic query and the set of hyperlinks in the first paragraph (Lesk, 1986). We choose the expansion for the entry with the highest overlap. If the query context does not help in disambiguation, we use the expansions for all partial matches that tie for the highest overlap.

3. If an article with the required name does not exist, the word will not be expanded.

2b. Query expansion with WordNet: Extract all nouns and NEs from the topic, and expand them with hypernyms, hyponyms and antonyms in WordNet 2.0:

1. If an word (or NE) W from the query corresponds to an unambiguous entry in WordNet, expand that entry.

2. If W has multiple senses, choose the sense(s) which have the highest overlap with the query. To compute overlap, for a sense we take its expansions (one step hypernyms, hyponyms and antonyms) and the words from the definition.

3. If W has no senses in WordNet, the word will not be expanded.

3. Expansion filtering: Filter the list of related concepts: keep only terms that appear in the document collection for the current topic.

Table 1 includes the expansions obtained from Wikipedia and from WordNet respectively for a number of words in topics from the DUC 2007 collection. *mining* is a specific activity, involving a limited set of materials. While such connections cannot be retrieved through hypernym, meronym or other semantic relations in WordNet, they are part of encyclopedic knowledge, and can be found in Wikipedia. *flight* is a more general concept – there are specific types of flight, which appear as hyponyms in WordNet, while in Wikipedia it is more generally described as the motion of an object through air, which does not provide us with interesting related concepts. *status* is a very general concept, and rather vague, for which neither WordNet nor Wikipedia can provide very useful information. Finally, Wikipedia is rich in named entities, which are not in the scope of a semantic lexicon. WordNet does contain named entities, but not on the scale on which Wikipedia does.

For the 45 topics from DUC 2007, the expansion with Wikipedia generated 1054 additional words, while with WordNet 2510. This difference comes from the fact that with Wikipedia it is mostly the NEs that are expanded, whereas with WordNet the common nouns, which are more numerous in the topics. The overlap between the two sets of expansions is 48 words (0.046 relative to Wikipedia expansions, 0.019 relative to WordNet).

4 Topic Expansion with Spreading Activation and PageRank

Concepts related to the ones in the topic provide a good handle on the documents to summarize – they indicate parts of the document that should be included in the summary. It is however obvious that the summary should contain more than that, and this information comes from the documents to be summarized. Amini & Usunier (2007) have shown that expanding the query within the set of documents leads to good results. Following this idea, to find more relevant concepts we look for words/NEs which are related to the topic, and at the same time important in the collection of documents for the given topic. The methods described in this section are applied on a large graph that covers the entire document collection for one topic. The documents are processed in a similar way to the query – parsed with the Stanford Parser, output in dependency relation format, lemmatized using XTag's morphological data file. The graph consists of nodes corresponding to lemmatized words and NEs in the documents, and edges correspoding to grammatical dependency relations.

4.1 Spreading Activation

To find words/NEs related to the topic we spread an activation signal starting from the topic words and their expansions (in a manner similar to (Mani & Bloedorn, 1999), and using an algorithm inspired by (Anderson, 1983)), which are given a node weight of 1. As we traverse the graph starting from these nodes, the signal is propagated by assigning a weight to each edge and each node traversed based on the signal strength. The signal strength diminishes with the distance from the node of origin depending on a signal decay parameter, according to the formula:

$$\begin{aligned} w_n(N_0) &= 1; \\ s_t &= (1 - decay) * \frac{w_n(N_t)}{Out(N_t)}; \\ w_n(N_{t+1}) &= s_t; \\ w_e(N_t, N_{t+1})_{t+1} &= w_e(N_t, N_{t+1})_t + s_t; \end{aligned}$$

where N_t is the current node; N_{t+1} is the node we are moving towards; $w_n(N_t)$ is the weight of node N_t; s_t is the signal strength at step t; $Out(N_t)$

Topic	Topic expanded words	Top ranked nodes
D0738 What is the status of mining in central and South America? Include obstacles encountered.	*status*, registered, *South America*, *central*, 1998, obstacle, *mining*, lead, agricultural, mineral, gold, ore, petroleum, nickel, iron, coal, tin, value, copper, water, bauxite, silver, diamond, *include*, *encounter*	company, dollar, project, sector, iron, mine, silver, percent, big, value, industry, source, overturn, regulate, link, official, decree, financing, expert, firm, activity, estimate, state, For Peru, Peru, third, already, top, 12th, creation, ton
D0717 Describe the various lawsuits against American Home Products which resulted from the use of fenfluramine, also known as Pondimin, and half of the diet drug combination called ”fen-phen”.	*combination*, set, *half*, *American Home Products*, *know*, *fenfluramine*, phentermine, obesity, release, dexfenfluramine, *use*, United States, Wal-Mart, *fen*, *describe*, *diet*, *call*, *drug*, drugs, medication, patients, medicine, *lawsuit*, right, court, damages, defendant, plaintiff, *also*, *various*, *Pondimin*, *result*	drug, market, company, settle, redux, claim, American Home Products, make, cause, seek, cover, people, allow, agree, dismiss, other, sue, case, Pondimin, state, link, million, award, user, estimate, thousand, file, think, note, damages, Harris County

Table 2: Top ranked nodes after expanding the topic with spreading activation and PageRank

is the number of outgoing edges from node N_t; $w_e(N_t, N_{t+1})_t$ is the weight of the edge between N_t and N_{t+1} at time t (i.e., before actually traversing the edge and spreading the activation from N_t); $w_e(N_t, N_{t+1})_{t+1}$ is the weight of the edge after spreading activation. The weight of the edges is cumulative, to gather strength from all signals that pass through the edge. Activation is spread sequentially from each node in the (expanded) topic.

The $decay$ parameter is used to control how far the influence of the starting nodes should reach – the lower the decay, the farther the signal can reach.

4.2 PageRank

The previous step has assigned weights to edges in the graph, such that higher weights are closer to topic and/or topic expanded words. After this initialization of the graph, we run a PageRank algorithm (Brin & Page, 1998) to determine more important nodes. By running this algorithm after initializing the graph edge weights, from the nodes that are closer to topic and topic expanded words we boost those that are more important in the documents.

The starting point of the PageRank algorithm is the graph with weighted edges obtained in the previous step. The node weights are initialized with 1 (the starting value does not matter). Analysis of the documents graph for several topics has revealed that there is a large highly interconnected structure, and many disconnected small (2-3 nodes) fragments. Page Rank will run on this dense core structure. The PageRank algorithm is guaranteed to converge if the graph is aperiodic and irreducible (Grimmett & Stirzaker, 1989). Aperiodicity implies that the greatest common divisor of the graph's cycles is 1 – this condition is met. Irreducibility of the graph means that it has no leaves, and there are no two nodes with the same set of neighbours. The remedy in such cases is to connect each leaf to all other nodes in the graph, and conflate nodes with the same set of neighbours.

Once the graph topology meets the PageRank convergence conditions, we run the algorithm. The original formula for computing the rank of a node at each iteration step is:

$$PR(n_i) = \frac{1-d}{N} + d \sum_{n_j \in Adj_{n_i}} \frac{PR(n_j)}{Out(n_j)}$$

where n_i is a node, d is the damping factor (usually $d = 0.85$ and this is the value we use as well), N is the number of nodes in the graph, $PR(n_i)$ is the rank of node n_i, Adj_{n_i} is the set of nodes adjacent to n_i, and $Out(n_j)$ is the number of outgoing edges from n_j (our graph is non-directed, so this number is the total number of edges with one end in n_j). We adjust this formula to reflect the weights of the edges, and the version used is the following:

$$PR(n_i) = \frac{1-d}{N} + d \sum_{n_j \in Adj_{n_i}} PR(n_j) w_{out}(n_j);$$

Expansion	ROUGE-2	ROUGE-SU4	BE
none	0.09270 (0.08785 - 0.09762)	0.14587 (0.14019 - 0.1514)	0.04958 (0.04559 - 0.05413)
WN$_{with\ WSD}$	0.09494 (0.09086 - 0.09900)	0.15295 (0.14897 - 0.15681)	0.04985 (0.04606 - 0.05350)
WN$_{no\ WSD}$	0.09596 (0.09189 - 0.09990)	0.15357 (0.14947 - 0.15741)	0.05173 (0.04794 - 0.05550)
Wiki	**0.10173** (0.09721 - 0.10608)	**0.15725** (0.15345 - 0.16130)	**0.05542** (0.05125 - 0.05967)
WN$_{no\ WSD}$ + Wiki	0.09604 (0.09228 - 0.09980)	0.15315 (0.14923 - 0.15694)	0.05292 (0.04912 - 0.05647)

Table 3: Comparison of topic expansion methods with 95% confidence intervals.

$$w_{out}(n_j) = \sum_{n_k \in Adj_{n_j}} w_e(n_k, n_j)$$

In Table 2 we show examples of top ranked nodes for several topics, extracted with this algorithm. The words in italics are keywords/phrases from the topic query, and the top ranked nodes are listed in decreasing order of their rank.

5 Summarization

The summarization method implemented is based on the idea that the entities or events mentioned in the query are somehow connected to each other, and the documents to be summarized contain information that allows us to make these connections. We use again the graph for all the documents in the collection related to one topic, built using the dependency relation representation of the texts. The nodes in this graph are words/NEs, and the links are grammatical relations.

We extract from this graph the subgraph that covers connections between all open class words/NEs in the topic or expanded topic query. Each edge in the extracted subgraph corresponds to a grammatical relation in a sentence of a document. We collect all sentences thus represented in the subgraph, and rerank them based on the number of edges they cover, and the occurrence of topic or expanded topic terms. We use the following formula to compute a sentence score:

$$\begin{aligned} Score(S) &= topicWords * w_{word} \\ &+ expandedWords * w_{expandedWord} \\ &+ topRankedWords * w_{topRankedWord} \\ &+ edgesCovered * w_{subgraphEdge} \\ &+ depRelation * w_{depRelation} \end{aligned}$$

w_{word}, $w_{expandedWord}$, $w_{topRankedWord}$, $w_{subgraphEdge}$ and $w_{depRelation}$ are weight parameters that give different importance to exact words from the topic, expanded words, top ranked words and edges covered in the extracted subgraph. During all experiments these parameters are fixed.[5]

To form the summary we traverse the ranked list of sentences starting with the highest ranked one, and add sentences to a summary, or delete from the existing summary, based on a simple lexical overlap measure. We stop when the desired summary length is reached – for DUC 2005–2007, 250 words (last sentence may be truncated to fill the summary up to the allowed word limit).

6 Evaluation

Experiments are run on DUC 2007 main summarization task data, for the last experiment we used the DUC 2005 and DUC 2006 data as well. Performance is evaluated in terms of ROUGE-2, ROUGE-SU4 and BE recall, following the methodology and using the same parameters as in the DUC summarization events.

We analyze several types of topic expansion: no expansion, WordNet, Wikipedia, and within document collection expansion using spreading activation and Page Rank. The spreading activation method has several parameters whose values must be determined.

We first compare the summaries produced with no topic expansion, WordNet (WN) and Wikipedia (Wiki) respectively. Table 3 shows the results in terms of ROUGE and BE recall on the DUC 2007 (main) data. Word sense disambiguation (WSD) for expansion with WordNet did not work very well, as evidenced by the lower results for disambiguated expansion (WN with WSD) compared to the non-

[5]The values used were set following a small number of experiments on DUC 2007 data, as the purpose was not to tune the system for best performance, but rather to study the impact of more interesting parameters, in particular expansion type, decay and node ranking. The values used are the following: $w_{word} = 5$, $w_{expandedWord} = 2.5$, $w_{topRankedWord} = 0.5$, $w_{subgraphEdge} = 2$, $w_{depRelation} = 0$.

disambiguated one. A better disambiguation algorithm may reverse the situation. Expanding a topic only with Wikipedia hyperlinks gives the best results. At the document level, the results are not as clear cut. Figure 3 shows a comparison in terms of ROUGE-SU4 recall scores at the document level of the Wikipedia and WN (no WSD) expansion methods, sorted in increasing order of the Wikipedia-based expansion scores. The points are connected to allow the reader to follow the results for each method.

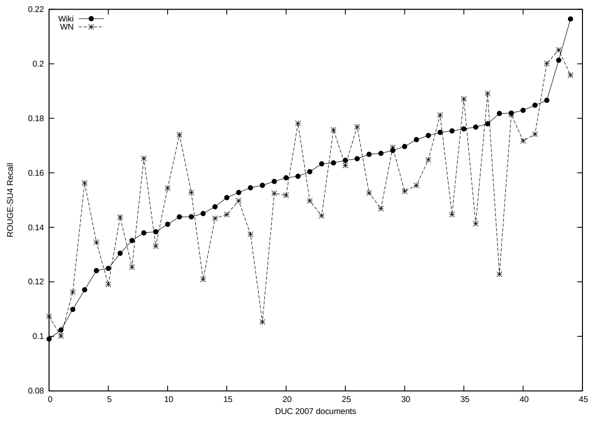


Figure 3: Comparison of Wikipedia and WN ROUGE-SU4 per-document recall results.

Because the overlap between Wikipedia and WordNet expanded queries was very low, we expected the two types of expansion to be complementary, and the combination to give better results than either expansion by itself. An analysis of results for each document with the three expansion methods – Wikipedia, WordNet, and their combination – showed that the simple combination of the expanded words cannot take advantage of the situations when one of the two methods performs better. In future work we will explore how to detect, based on the words in the query, which type of expansion is best, and how to combine them using a weighting scheme.

We choose the best configuration from above (Wikipedia expansion), and further expand the query through spreading activation and PageRank. This new type of expansion has two main parameters which influence the summarization outcome: number of top ranked nodes to add to the topic expansion, and the decay of the spreading activation algorithm.

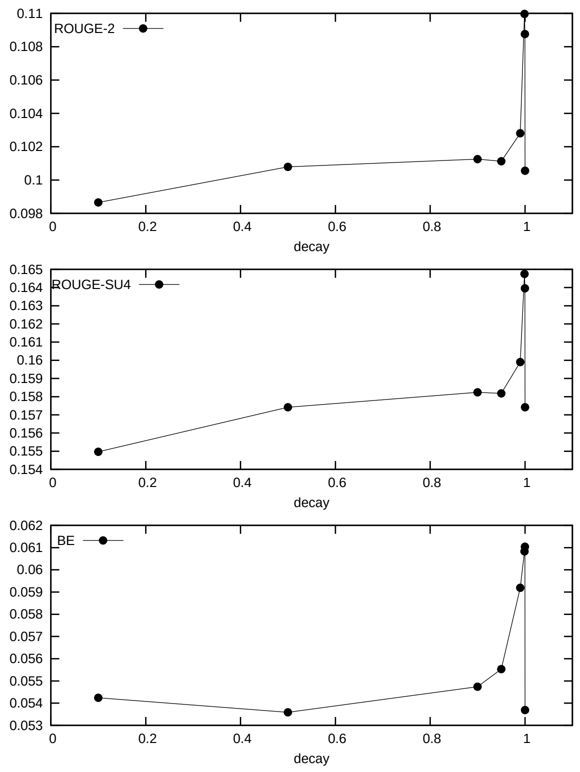


Figure 4: Impact of signal decay in spreading activation on summarization performance.

The decay parameter determines how far the influence of the starting nodes (words from query or Wikipedia-expanded query) should be felt. The results in Figure 4 – for decay values 0.1, 0.5, 0.95, 0.99, 0.999, 0.9999, 1 – indicate that faster decay (reflected through a higher decay value) keeps the summary more focused around the given topic, and leads to better results.[6] For a high enough decay – and eventually a decay of 1 – the weights of the edges become extremely small, and due to real number representation in memory, practically 0. In this situation PageRank has no effect, and all nodes have the same rank.

We fix the decay parameter to 0.9999, and we study the impact of the number of top nodes chosen after ranking with PageRank. Figure 5 shows the results when the number of top ranked nodes chosen varies. Adding highly ranked nodes benefits the performance of the system only up to a certain limit.

[6]During this set of experiments all other parameters are fixed, the number of top ranked nodes added to the topic expansion is 30.

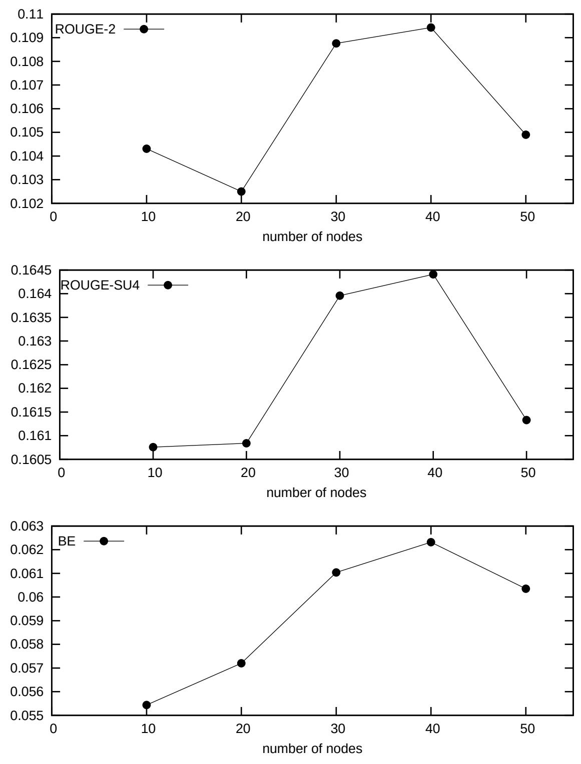


Figure 5: Impact of the number of top ranked nodes added to the expanded topic on summarization performance.

From the values we tested, the best results were obtained when adding 40 nodes to the expanded topic.

The best system configuration from the ones explored[7] is run on the DUC 2005, 2006 and 2007 (main) data. The performance and rank (in parentheses) compared to participating systems is presented in Table 4.

DUC	ROUGE-2	ROUGE-SU4	BE
2005 (32)	0.07074 (3)	0.13002 (2)	–
2006 (35)	0.08091 (11)	0.14022 (9)	0.04223 (11)
2007 (32)	0.11048 (6)	0.16479 (5)	0.06250 (5)

Table 4: System performance (and rank) on the DUC 2005, 2006 and 2007 (main) data. The number in parenthesis after the DUC year indicates the number of competing systems.

[7]Wikipedia expansion + 40 top nodes after spreading activation and PageRank, decay = 0.9999, $w_{expandedWord} = 3.5$, $w_{depRelation} = 1$, the other parameters have the same values as before.

7 Conclusions

The experiments conducted within the summarization framework of the Document Understanding Conference have confirmed that encyclopedic knowledge extracted from Wikipedia can benefit the summarization task. Wikipedia articles are a source of relevant related concepts, that are useful for expanding a summarization query. Furthermore, including information from the documents to be summarized by choosing relevant concepts – based on closeness to topic keywords and relative importance – improves even more the quality of the summaries, judged through ROUGE-2, ROUGE-SU4 and BE recall scores, as it is commonly done in the DUC competitions. The topic expansion methods explored lead to high summarization performance – ranked 2^{nd}, 9^{th} and 5^{th} on DUC 2005, 2006 and 2007 respectively according to ROUGE-SU4 scores – compared to (more than 30) DUC participating systems.

The graph representation of the documents is central to the summarization method we described. Because of this, we plan to improve this representation by collapsing together coreferential nodes and clustering together related concepts, and verify whether such changes impact the summarization results, as we expect they would.

Being able to move away from the topic within the set of documents and discover new relevant nodes is an important issue, especially from the point of view of a new summarization style – updates. In update summaries the starting point is a topic, which a summarization system must track in consecutive sets of documents. We can adjust the spreading activation parameters to how far a new set of documents is from the topic. Future work includes testing the spreading activation and page ranking method in the context of the update summarization task and exploring methods of extracting related concepts from the full text of Wikipedia articles.

Acknowledgments This work was funded by the Klaus Tschira Foundation, Heidelberg, Germany. We thank the anonymous reviewers for insightful comments and suggestions.

References

Ahn, D., V. Jijkoun, G. Mishne, K. Müller, M. de Rijke & S. Schlobach (2004). Using Wikipedia at the TREC QA track. In *Proc. of TREC-13*.

Amini, M. R. & N. Usunier (2007). A contextual query expansion approach by term clustering for robust text summarization. In *Proc. of DUC-07*.

Anderson, J. R. (1983). A spreading activation theory of memory. *Journal of Verbal Learning and Verbal Behaviour*, 22:261–295.

Auer, S., C. Bizer, J. Lehmann, G. Kobilarov, R. Cyganiak & Z. Ives (2007). DBpedia: A nucleus for a Web of open data. In *Proc. of ISWC 2007 + ASWC 2007*, pp. 722–735.

Barzilay, R. & M. Elhadad (1999). Using lexical chains for text summarization. In I. Mani & M. T. Maybury (Eds.), *Advances in Automatic Text Summarization*, pp. 111–121. Cambridge, Mass.: MIT Press.

Brin, S. & L. Page (1998). The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117.

Collins, A. M. & E. F. Loftus (1975). A spreading-activation theory of semantic processing. *Psychological Review*, (82):407–428.

Edmundson, H. (1969). New methods in automatic extracting. *Journal of the Association for Computing Machinery*, 16(2):264–285.

Erkan, G. & D. R. Radev (2004). LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Fellbaum, C. (Ed.) (1998). *WordNet: An Electronic Lexical Database.* Cambridge, Mass.: MIT Press.

Gabrilovich, E. & S. Markovitch (2006). Overcoming the brittleness bottleneck using Wikipedia: Enhancing text categorization with encyclopedic knowledge. In *Proc. of AAAI-06*, pp. 1301–1306.

Gotti, F., G. Lapalme, L. Nerima & E. Wehrli (2007). GOFAIsum: a symbolic summarizer for DUC. In *Proc. of DUC-07*.

Grimmett, G. & D. Stirzaker (1989). *Probability and Random Processes.* Oxford University Press.

Harabagiu, S. (2004). Incremental topic representations. In *Proc. of COLING-04*, pp. 583–589.

Hickl, A., K. Roberts & F. L. C. C. Lacatusu (2007). LCC's GISTexter at DUC 2007: Machine reading for update summarization. In *Proc. of DUC-07*.

Katz, B., G. Marton, G. Borchardt, A. Brownell, S. Felshin, D. Loreto, J. Louis-Rosenberg, B. Lu, F. Mora, S. Stiller, O. Uzuner & A. Wilcox (2005). External knowledge sources for Question Answering. In *Proc. of TREC-14*.

Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proc. of ACSD-86*, pp. 24–26.

Leskovec, J., M. Grobelnik & N. Milic-Frayling (2004). Learning sub-structures of document semantic graphs for document summarization. In *Proc. of LinkKDD-04*.

Lin, C.-Y. & E. Hovy (2000). The automated acquisition of topic signatures for automatic summarization. In *Proc. of COLING-00*, pp. 495–501.

Mani, I. & E. Bloedorn (1999). Summarizing similarities and differences among related documents. *Information Retrieval*, 1(1):35–67.

Mihalcea, R. & P. Tarau (2004). TextRank: Bringing order into texts. In *Proc. EMNLP-04*, pp. 404–411.

Mohamed, A. A. & S. Rajasekaran (2006). Query-based summarization based on document graphs. In *Proc. of DUC-06*.

Ponzetto, S. P. & M. Strube (2007a). Deriving a large scale taxonomy from Wikipedia. In *Proc. of AAAI-07*, pp. 1440–1445.

Ponzetto, S. P. & M. Strube (2007b). Knowledge derived from Wikipedia for computing semantic relatedness. *Journal of Artificial Intelligence Research*, 30:181–212.

Quillian, M. R. (1967). Word concepts: A theory and simulation of some basic semantic capabilities. *Behavioural Science*, 12(5):410–430.

Rath, G., A. Resnick & T. Savage (1961). The formation of abstracts by the selection of sentences. *American Documentation*, 12(2):139–143.

Suchanek, F. M., G. Kasneci & G. Weikum (2007). YAGO: A core of semantic knowledge. In *Proc. of WWW-07*, pp. 697–706.

Ye, S. & T.-S. Chua (2006). NUS at DUC 2006: Document concept lattice for summarization. In *Proc. of DUC-06*.

Summarizing Spoken and Written Conversations

Gabriel Murray and **Giuseppe Carenini**
Department of Computer Science
University of British Columbia
Vancouver, BC V6T 1Z4 Canada

Abstract

In this paper we describe research on summarizing conversations in the meetings and emails domains. We introduce a conversation summarization system that works in multiple domains utilizing general conversational features, and compare our results with domain-dependent systems for meeting and email data. We find that by treating meetings and emails as conversations with general conversational features in common, we can achieve competitive results with state-of-the-art systems that rely on more domain-specific features.

1 Introduction

Our lives are increasingly comprised of multimodal conversations with others. We email for business and personal purposes, attend meetings in person and remotely, chat online, and participate in blog or forum discussions. It is clear that automatic summarization can be of benefit in dealing with this overwhelming amount of interactional information. Automatic meeting abstracts would allow us to prepare for an upcoming meeting or review the decisions of a previous group. Email summaries would aid corporate memory and provide efficient indices into large mail folders.

When summarizing in each of these domains, there will be potentially useful domain-specific features – e.g. prosodic features for meeting speech, subject headers for emails – but there are also underlying similarites between these domains. They are all multiparty conversations, and we hypothesize that effective summarization techniques can be designed that would lead to robust summarization performance on a wide array of such conversation types. Such a general conversation summarization system would make it possible to summarize a wide variety of conversational data without needing to develop unique summarizers in each domain and across modalities. While progress has been made in summarizing conversations in individual domains, as described below, little or no work has been done on summarizing unrestricted, multimodal conversations.

In this research we take an extractive approach to summarization, presenting a novel set of conversational features for locating the most salient sentences in meeting speech and emails. We demonstrate that using these conversational features in a machine-learning sentence classification framework yields performance that is competitive or superior to more restricted domain-specific systems, while having the advantage of being portable across conversational modalities. The robust performance of the conversation-based system is attested via several summarization evaluation techniques, and we give an in-depth analysis of the effectiveness of the individual features and feature subclasses used.

2 Related Work on Meetings and Emails

In this section we give a brief overview of previous research on meeting summarization and email summarization, respectively.

2.1 Meeting Summarization

Among early work on meeting summarization, Waibel et al. (1998) implemented a modified version of the Maximal Marginal Relevance algorithm (Carbonell and Goldstein, 1998) applied to speech transcripts, presenting the user with the *n* best sentences in a meeting browser interface. Zechner (2002) investigated summarizing several genres of speech, including spontaneous meeting speech. Though relevance detection in his work relied largely on *tf.idf* scores, Zechner also explored cross-speaker information linking and question/answer detection.

More recently, researchers have investigated the utility of employing speech-specific features for summarization, including prosodic information. Murray et al. (2005a; 2005b) compared purely textual summarization approaches with feature-based approaches incorporating prosodic features, with human judges favoring the feature-based approaches. In subsequent work (2006; 2007), they began to look at additional speech-specific characteristics such as speaker status, discourse markers and high-level meta comments in meetings, i.e. comments that refer to the meeting itself. Galley (2006) used skip-chain Conditional Random Fields to model pragmatic dependencies between paired meeting utterances (e.g. QUESTION-ANSWER relations), and used a combination of lexical, prosodic, structural and discourse features to rank utterances by importance. Galley found that while the most useful single feature class was *lexical* features, a combination of acoustic, durational and structural features exhibited comparable performance according to Pyramid evaluation.

2.2 Email Summarization

Work on email summarization can be divided into summarization of individual email messages and summarization of entire email threads. Muresan et al. (2001) took the approach of summarizing individual email messages, first using linguistic techniques to extract noun phrases and then employing machine learning methods to label the extracted noun phrases as salient or not. Corston-Oliver et al. (2004) focused on identifying speech acts within a given email, with a particular interest in task-related sentences.

Rambow et al. (2004) addressed the challenge of summarizing entire threads by treating it as a binary sentence classification task. They considered three types of features: basic features that simply treat the email as text (e.g. *tf.idf*, which scores words highly if they are frequent in the document but rare across all documents), features that consider the thread to be a sequence of turns (e.g. the position of the turn in the thread), and email-specific features such as number of recipients and subject line similarity.

Carenini et al. (2007) took an approach to thread summarization using the Enron corpus (described below) wherein the thread is represented as a fragment quotation graph. A single node in the graph represents an *email fragment*, a portion of the email that behaves as a unit in a fine-grain representation of the conversation structure. A fragment sometimes consists of an entire email and sometimes a portion of an email. For example, if a given email has the structure

A
$>$ B
C

where B is a quoted section in the middle of the email, then there are three email fragments in total: two new fragments A and C separated by one quoted fragment B. Sentences in a fragment are weighted according to the Clue Word Score (CWS) measure, a lexical cohesion metric based on the recurrence of words in parent and child nodes. In subsequent work, Carenini et al. (2008) determined that subjectivity detection (i.e., whether the sentence contains sentiments or opinions from the author) gave additional improvement for email thread summaries.

Also on the Enron corpus, Zajic et al. (2008) compared Collective Message Summarization (CMS) to Individual Message Summarization (IMS) and found the former to be a more effective technique for summarizing email data. CMS essentially treats thread summarization as a multi-document summarization problem, while IMS summarizes individual emails in the thread and then concatenates them to form a thread summary.

In our work described below we also address the task of thread summarization as opposed to summarization of individual email messages, following Carenini et al. and the CMS approach of Zajic et al.

3 Experimental Setup

In this section we describe the classifier employed for our machine learning experiments, the corpora used, the relevant summarization annotations for each corpus, and the evaluation methods employed.

3.1 Statistical Classifier

Our approach to extractive summarization views sentence extraction as a classification problem. For all machine learning experiments, we utilize logistic regression classifiers. This choice was partly motivated by our earlier summarization research, where logistic regression classifiers were compared alongside support vector machines (SVMs) (Cortes and Vapnik, 1995). The two classifier types yielded very similar results, with logistic regression classifiers being much faster to train and thus expediting further development.

The *liblinear* toolkit [1] implements simple feature subset selection based on the F statistic (Chen and Lin, 2006) .

3.2 Corpora Description

For these experiments we utilize two corpora, the Enron corpus for email summarization and the AMI corpus for meeting summarization.

3.2.1 The Enron Email Corpus

The Enron email corpus[2] is a collection of emails released as part of the investigation into the Enron corporation (Klimt and Yang, 2004). It has become a popular corpus for NLP research (e.g. (Bekkerman et al., 2004; Yeh and Harnly, 2006; Chapanond et al., 2005; Diesner et al., 2005)) due to being realistic, naturally-occurring data from a corporate environment, and moreover because privacy concerns mean that there is very low availability for other publicly available email data.

39 threads have been annotated for extractive summarization, with five annotators assigned to each thread. The annotators were asked to select 30% of the sentences in a thread, subsequently labeling each selected sentence as either ’essential’ or ’optional.’ Essential sentences are weighted three times as highly as optional sentences. A sentence score, or GSValue, can therefore range between 0 and 15, with the maximum GSValue achieved when all five annotators consider the sentence essential, and a score of 0 achieved when no annotator selects the given sentence. For the purpose of training a binary classifier, we rank the sentences in each email thread according to their GSValues, then extract sentences until our summary reaches 30% of the total thread word count. We label these sentences as positive instances and the remainder as the negative class. Approximately 19% of sentences are labeled as positive, extractive examples.

Because the amount of labeled data available for the Enron email corpus is fairly small, for our classification experiments we employ a leave-one-out proceedure for the 39 email threads. The labeled data as a whole total just under 1400 sentences.

3.2.2 The AMI Meetings Corpus

For our meeting summarization experiments, we use the *scenario* portion of the AMI corpus (Carletta et al., 2005). The corpus consists of about 100 hours of recorded and annotated meetings. In the scenario meetings, groups of four participants take part in a series of four meetings and play roles within a fictitious company. While the scenario given to them is artificial, the speech and the actions are completely spontaneous and natural. There are 96 meetings in the training set, 24 in the development set, and 20 meetings for the test set.

For this corpus, annotators wrote abstract summaries of each meeting and extracted transcript dialogue act segments (DAs) that best conveyed or supported the information in the abstracts. A many-to-many mapping between transcript DAs and sentences from the human abstract was obtained for each annotator, with three annotators assigned to each meeting. It is possible for a DA to be extracted by an annotator but not linked to the abstract, but for training our binary classifiers, we simply consider a dialogue act to be a positive example if it is linked to a given human summary, and a negative example otherwise. This is done to maximize the likelihood that a data point labeled as “extractive” is truly an informative example for training purposes. Approximately 13% of the total DAs are ultimately labeled

[1]http://www.csie.ntu.edu.tw/˜cjlin/liblinear/

[2]http://www.cs.cmu.edu/˜enron/

as positive, extractive examples.

The AMI corpus contains automatic speech recognition (ASR) output in addition to manual meeting transcripts, and we report results on both transcript types. The ASR output was provided by the AMI-ASR team (Hain et al., 2007), and the word error rate for the AMI corpus is 38.9%.

3.3 Summarization Evaluation

For evaluating our extractive summaries, we implement existing evaluation schemes from previous research, with somewhat similar methods for meetings versus emails. These are described and compared below. We also evaluate our extractive classifiers more generally by plotting the receiver operator characteristic (ROC) curve and calculating the area under the curve (AUROC). This allows us to gauge the true-positive/false-positive ratio as the posterior threshold is varied.

We use the differing evaluation metrics for emails versus meetings for two primary reasons. First, the differing summarization annotations in the AMI and Enron corpora naturally lend themselves to slightly divergent metrics, one based on extract-abstract links and the other based on the essential/option/uninformative distinction. Second, and more importantly, using these two metrics allow us to compare our results with state-of-the-art results in the two fields of speech summarization and email summarization. In future work we plan to use a single evaluation metric.

3.3.1 Evaluating Meeting Summaries

To evaluate meeting summaries we use the weighted f-measure metric (Murray et al., 2006). This evaluation scheme relies on the multiple human annotated summary links described in Section 3.2.2. Both weighted precision and recall share the same numerator

$$num = \sum_{i=1}^{M} \sum_{j=1}^{N} L(s_i, a_j) \qquad (1)$$

where $L(s_i, a_j)$ is the number of links for a DA s_i in the machine extractive summary according to annotator a_i, M is the number of DAs in the machine summary, and N is the number of annotators. Weighted precision is defined as:

$$precision = \frac{num}{N \cdot M} \qquad (2)$$

and weighted recall is given by

$$recall = \frac{num}{\sum_{i=1}^{O} \sum_{j=1}^{N} L(s_i, a_j)} \qquad (3)$$

where O is the total number of DAs in the meeting, N is the number of annotators, and the denominator represents the total number of links made between DAs and abstract sentences by all annotators. The weighted f-measure is calculated as the harmonic mean of weighted precision and recall. The intuition behind weighted f-score is that DAs that are linked multiple times by multiple annotators are the most informative.

3.3.2 Evaluating Email Summaries

For evaluating email thread summaries, we follow Carenini et al. (2008) by implementing their *pyramid precision* scheme, inspired by Nenkova's pyramid scheme (2004). In Section 3.2.1 we introduced the idea of a GSValue for each sentence in an email thread, based on multiple human annotations. We can evaluate a summary of a given length by comparing its total GSValues to the maximum possible total for that summary length. For instance, if in a thread the three top scoring sentences had GSValues of 15, 12 and 12, and the sentences selected by a given automatic summarization method had GSValues of 15, 10 and 8, the pyramid precision would be 0.85.

Pyramid precision and weighted f-score are similar evaluation schemes in that they are both sentence based (as opposed to, for example, n-gram based) and that they score sentences based on multiple human annotations. Pyramid precision is very similar to equation 3 normalized by the maximum score for the summary length. For now we use these two slightly different schemes in order to maintain consistency with prior art in each domain.

4 A Conversation Summarization System

In our conversation summarization approach, we treat emails and meetings as conversations comprised of turns between multiple participants. We follow Carenini et al. (2007) in working at the finer granularity of email fragments, so that for an email thread, a turn consists of a single email fragment in the exchange. For meetings, a turn is a sequence of dialogue acts by one speaker, with the turn boundaries delimited by dialogue acts from other meeting participants. The features we derive for summarization are based on this view of the conversational structure.

We calculate two **length** features. For each sentence, we derive a word-count feature normalized by the longest sentence in the conversation (*SLEN*) and a word-count feature normalized by the longest sentence in the turn (*SLEN2*). Sentence length has previously been found to be an effective feature in speech and text summarization (e.g. (Maskey and Hirschberg, 2005; Murray et al., 2005a; Galley, 2006)).

There are several **structural** features used, including position of the sentence in the turn (*TLOC*) and position of the sentence in the conversation (*CLOC*). We also include the time from the beginning of the conversation to the current turn (*TPOS1*) and from the current turn to the end of the conversation (*TPOS2*). Conversations in both modalities can be well-structured, with introductory turns, general discussion, and ultimate resolution or closure, and sentence informativeness might significantly correlate with this structure. We calculate two pause-style features: the time between the following turn and the current turn (*SPAU*), and the time between the current turn and previous turn (*PPAU*), both normalized by the overall length of the conversation. These features are based on the email and meeting transcript timestamps. We hypothesize that pause features may be useful if informative turns tend to elicit a large number of responses in a short period of time, or if they tend to quickly follow a preceding turn, to give two examples.

There are two features related to the conversation **participants** directly. One measures how dominant the current participant is in terms of words in the conversation (*DOM*), and the other is a binary feature indicating whether the current participant initiated the conversation (*BEGAUTH*), based simply on whether they were the first contributor. It is hypothesized that informative sentences may more often belong to participants who lead the conversation or have a good deal of dominance in the discussion.

There are several **lexical** features used in these experiments. For each unique word, we calculate two conditional probabilities. For each conversation participant, we calculate the probability of the participant given the word, estimating the probability from the actual term counts, and take the maximum of these conditional probabilities as our first term score, which we will call *Sprob*.

$$Sprob(t) = \max_S p(S|t)$$

where t is the word and S is a participant. For example, if the word *budget* is used ten times in total, with seven uses by participant A, three uses by participant B and no uses by the other participants, then the *Sprob* score for this term is 0.70. The intuition is that certain words will tend to be associated with one conversation participant more than the others, owing to varying interests and expertise between the people involved.

Using the same procedure, we calculate a score called *Tprob* based on the probability of each turn given the word.

$$Tprob(t) = \max_T p(T|t)$$

The motivating factor for this metric is that certain words will tend to cluster into a small number of turns, owing to shifting topics within a conversation.

Having derived *Sprob* and *Tprob*, we then calculate several sentence-level features based on these term scores. Each sentence has features related to max, $mean$ and sum of the term scores for the words in that sentence (*MXS*, *MNS* and *SMS* for *Sprob*, and *MXT*, *MNT* and *SMT* for *Tprob*). Using a vector representation, we calculate the cosine between the conversation preceding the given sentence and the conversation subsequent to the sentence, first using *Sprob* as the vector weights (*COS1*) and then using *Tprob* as the vector weights (*COS2*). This is motivated by the hypothesis that informative sentences might change the conversation in some fashion, leading to a low cosine between the preceding and subsequent portions. We similarly calculate two scores measuring the cosine between the current sentence and the rest of the converation, using each term-weight metric as vector weights (*CENT1* for *Sprob* and *CENT2* for *Tprob*). This measures

Feature ID	Description
MXS	max *Sprob* score
MNS	mean *Sprob* score
SMS	sum of *Sprob* scores
MXT	max *Tprob* score
MNT	mean *Tprob* score
SMT	sum of *Tprob* scores
TLOC	position in turn
CLOC	position in conv.
SLEN	word count, globally normalized
SLEN2	word count, locally normalized
TPOS1	time from beg. of conv. to turn
TPOS2	time from turn to end of conv.
DOM	participant dominance in words
COS1	cos. of conv. splits, w/ *Sprob*
COS2	cos. of conv. splits, w/ *Tprob*
PENT	entro. of conv. up to sentence
SENT	entro. of conv. after the sentence
THISENT	entropy of current sentence
PPAU	time btwn. current and prior turn
SPAU	time btwn. current and next turn
BEGAUTH	is first participant (0/1)
CWS	rough ClueWordScore
CENT1	cos. of sentence & conv., w/ *Sprob*
CENT2	cos. of sentence & conv., w/ *Tprob*

Table 1: Features Key

whether the candidate sentence is generally similar to the conversation overall.

There are three word entropy features, calculated using the formula

$$went(s) = \frac{\sum_{i=1}^{N} p(x_i) \cdot -\log(p(x_i))}{(\frac{1}{N} \cdot -\log(\frac{1}{N})) \cdot M}$$

where s is a string of words, x_i is a word type in that string, $p(x_i)$ is the probability of the word based on its normalized frequency in the string, N is the number of word types in the string, and M is the number of word tokens in the string.

Note that word entropy essentially captures information about type-token ratios. For example, if each word token in the string was a unique type then the word entropy score would be 1. We calculate the word entropy of the current sentence (*THISENT*), as well as the word entropy for the conversation up until the current sentence (*PENT*) and the word entropy for the conversation subsequent to the current sentence (*SENT*). We hypothesize that informative sentences themselves may have a diversity of word types, and that if they represent turning points in the conversation they may affect the entropy of the subsequent conversation.

Finally, we include a feature that is a rough approximation of the ClueWordScore (CWS) used by Carenini et al. (2007). For each sentence we remove stopwords and count the number of words that occur in other turns besides the current turn. The CWS is therefore a measure of conversation cohesion.

For ease of reference, we hereafter refer to this conversation features system as ConverSumm.

5 Comparison Summarization Systems

In order to compare the ConverSumm system with state-of-the-art systems for meeting and email summarization, respectively, we also present results using the features described by Murray and Renals (2008) for meetings and the features described by Rambow (2004) for email. Because the work by Murray and Renals used the same dataset, we can compare our scores directly. However, Rambow carried out summarization work on a different, unavailable email corpus, and so we re-implemented their summarization system for our current email data.

In their work on meeting summarization, Murray and Renals creating 700-word summaries of each meeting using several classes of features: prosodic, lexical, structural and speaker-related. While there are two features overlapping between our systems (word-count and speaker/participant dominance), their system is primarily domain-dependent in its use of prosodic features while our features represent a more general conversational view.

Rambow presented 14 features for the summarization task, including email-specific information such as the number of recipients, number of responses, and subject line overlap. There is again a slight overlap in features between our two systems, as we both include length and position of the sentence in the thread/conversation.

6 Results

Here we present, in turn, the summarization results for meeting and email data.

6.1 Meeting Summarization Results

Figure 1 shows the F statistics for each Conversumm feature in the meeting data, providing a measure of the usefulness of each feature in discriminating between the positive and negative classes. Some

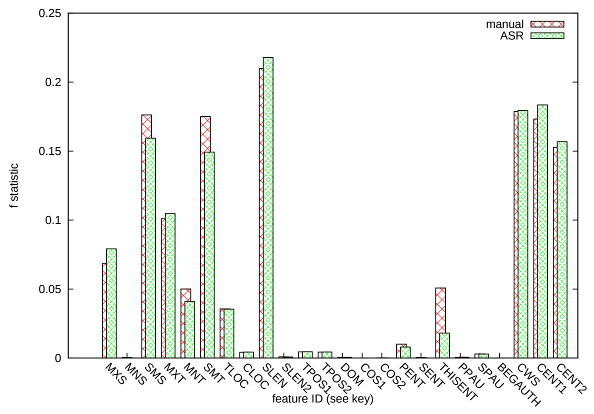


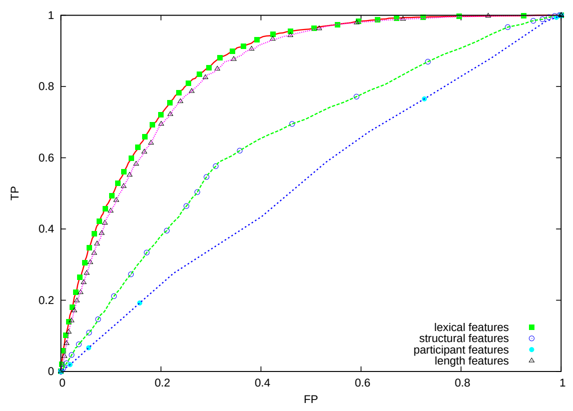
Figure 1: Feature F statistics for AMI meeting corpus

System	Weighted F-Score	AUROC
Speech - Man	0.23	0.855
Speech - ASR	0.24	0.850
Conv. - Man	0.23	0.852
Conv. - ASR	0.22	0.853

Table 2: Weighted F-Scores and AUROCs for Meeting Summaries

features such as participant dominance have very low F statistics because each sentence by a given participant will receive the same score; so while the feature itself may have a low score because it does not discriminate informative versus non-informative sentences on its own, it may well be useful in conjunction with the other features. The best individual ConverSumm features for meeting summarization are sentence length (SLEN), sum of $Sprob$ scores, sum of $Tprob$ scores, the simplified CWS score (CWS), and the two centroid measures (CENT1 and CENT2). The word entropy of the candidate sentence is very effective for manual transcripts but much less effective on ASR output. This is due to the fact that ASR errors can incorrectly lead to high entropy scores.

Table 2 provides the weighted f-scores for all summaries of the meeting data, as well as AUROC scores for the classifiers themselves. For our 700-word summaries, the Conversumm approach scores comparably to the speech-specific approach on both manual and ASR transcripts according to weighted f-score. There are no significant differences according to paired t-test. For the AUROC measures, there are again no significant differences between the conversation summarizers and speech-specific summarizers. The AUROC for the conversation system is slightly lower on manual transcripts and slightly higher when applied to ASR output.



Fea. Subset	AUROC
Structural	0.652
Participant	0.535
Length	0.837
Lexical	0.852

Figure 2: AUROC Values for Feature Subclasses, AMI Corpus

For all systems the weighted f-scores are somewhat low. This is partly owing to the fact that output summaries are very short, leading to high precision and low recall. The low f-scores are also indicative of the difficulty of the task. Human performance, gauged by comparing each annotator's summaries to the remaining annotators' summaries, exhibits an average weighted f-score of 0.47 on the same test set. The average kappa value on the test set is 0.48, showing the relatively low inter-annotator agreement that is typical of summarization annotation. There is no additional benefit to combining the conversational and speech-specific features. In that case, the weighted f-scores are 0.23 for both manual and ASR transcripts. The overall AUROC is 0.85 for manual transcripts and 0.86 for ASR.

We can expand the features analysis by considering the effectiveness of certain subclasses of features. Specifically, we group the summarization features into *lexical*, *structural*, *participant* and *length* features. Figure 2 shows the AUROCs for the feature subset classifiers, illustrating that the lexical subclass is very effective while the length features also constitute a challenging baseline. A weakness

System	Pyramid Precision	AUROC
Rambow	0.50	0.64
Conv.	0.46	0.75

Table 3: Pyramid Precision and AUROCs for Email Summaries

of systems that depend heavily on length features, however, is that recall scores tend to decrease because the extracted units are much longer - weighted recall scores for the 700 word summaries are significantly worse according to paired t-test ($p<0.05$) when using just length features compared to the full feature set.

6.2 Email Summarization Results

Figure 3 shows the F statistic for each ConverSumm feature in the email data.The two most useful features are sentence length and CWS. The *Sprob* and *Tprob* features rate very well according to the F statistic. The two centroid features incorporating *Sprob* and *Tprob* are comparable to one another and are very effective features as well.

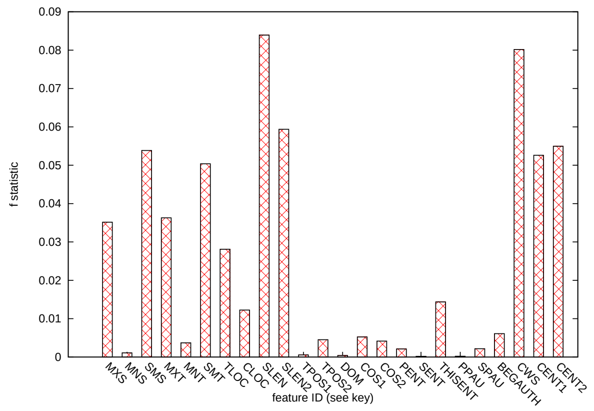
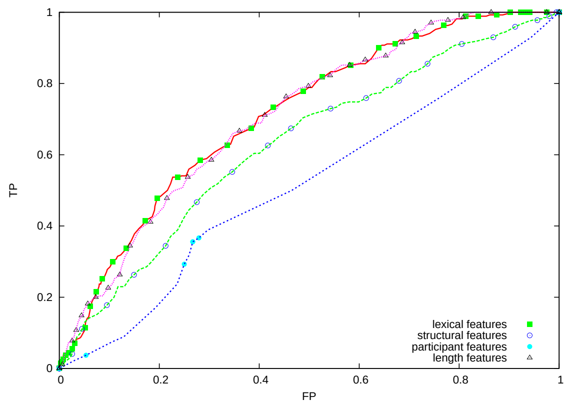


Figure 3: Feature F statistics for Enron email corpus

After creating 30% word compression summaries using both the ConverSumm and Rambow approaches, we score the 39 thread summaries using Pyramid Precision. The results are given in Table 3. On average, the Rambow system is slightly higher with a score of 0.50 compared with 0.46 for the conversational system, but there is no statistical difference according to paired t-test.

The average AUROC for the Rambow system is 0.64 compared with 0.75 for the ConverSumm system, with ConverSumm system significantly better according to paired t-test ($p<0.05$). Random classification performance would yield an AUROC of 0.5.



Fea. Subset	AUROC
Structural	0.63
Participant	0.51
Length	0.71
Lexical	0.71

Figure 4: AUROC Values for Feature Subclasses, Enron Corpus

Combining the Rambow and ConverSumm features does not yield any overall improvement. The Pyramid Precision score in that case is 0.47 while the AUROC is 0.74.

Figure 4 illustrates that the lexical and length features are the most effective feature subclasses, though the best results overall are derived from a combination of all feature classes.

7 Discussion

According to multiple evaluations, the ConverSumm features yield competitive summarization performance with the comparison systems. There is a clear set of features that are similarly effective in both domains, especially CWS, the centroid features, the $Sprob$ features, the $Tprob$ features, and sentence length. There are other features that are more effective in one domain than the other. For example, the BEGAUTH feature, indicating whether the current participant began the conversation, is more useful for emails. It seems that being the first person to speak in a meeting is not as significant as being the first person to email in a given thread. SLEN2, which normalizes sentence length by the longest sentence in the turn, also is much more effective for emails. The reason is that many meeting turns consist of a single, brief utterance such as "Okay, yeah."

The finding that the summary evaluations are not significantly worse on noisy ASR compared with manual transcripts has been previously attested (Valenza et al., 1999; Murray et al., 2005a), and it is encouraging that our ConverSumm features are similarly robust to this noisy data.

8 Conclusion

We have shown that a general conversation summarization approach can achieve results on par with state-of-the-art systems that rely on features specific to more focused domains. We have introduced a conversation feature set that is similarly effective in both the meetings and emails domains. The use of multiple summarization evaluation techniques confirms that the system is robust, even when applied to the noisy ASR output in the meetings domain. Such a general conversation summarization system is valuable in that it may save time and effort required to implement unique systems in a variety of conversational domains.

We are currently working on extending our system to other conversation domains such as chats, blogs and telephone speech. We are also investigating domain adaptation techniques; for example, we hypothesize that the relatively well-resourced domain of meetings can be leveraged to improve email results, and preliminary findings are encouraging.

References

R. Bekkerman, A. McCallum, and G. Huang. 2004. Automatic categorization of email into folders: Benchmark experiments on Enron and SRI corpora. Technical Report IR-418, Center of Intelligent Information Retrieval, UMass Amherst.

J. Carbonell and J. Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proc. of ACM SIGIR Conference on Research and Development in Information Retrieval 1998, Melbourne, Australia*, pages 335–336.

G. Carenini, R. Ng, and X. Zhou. 2007. Summarizing email conversations with clue words. In *Proc. of ACM WWW 07, Banff, Canada*.

G. Carenini, X. Zhou, and R. Ng. 2008. Summarizing emails with conversational cohesion and subjectivity. In *Proc. of ACL 2008, Columbus, Ohio, USA*.

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 *Proc. of MLMI 2005, Edinburgh, UK*, pages 28–39.

A. Chapanond, M. Krishnamoorthy, and B. Yener. 2005. Graph theoretic and spectral analysis of enron email data. *Comput. Math. Organ. Theory*, 11(3):265–281.

Y-W. Chen and C-J. Lin. 2006. Combining SVMs with various feature selection strategies. In I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, editors, *Feature extraction, foundations and applications*. Springer.

S. Corston-Oliver, E. Ringger, M. Gamon, and R. Campbell. 2004. Integration of email and task lists. In *Proc. of CEAS 2004, Mountain View, CA, USA*.

C. Cortes and V. Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297.

J. Diesner, T. Frantz, and K. Carley. 2005. Communication networks from the enron email corpus "it's always about the people. enron is no different". *Comput. Math. Organ. Theory*, 11(3):201–228.

M. Galley. 2006. A skip-chain conditional random field for ranking meeting utterances by importance. In *Proc. of EMNLP 2006, Sydney, Australia*, pages 364–372.

T. Hain, L. Burget, J. Dines, G. Garau, V. Wan, M. Karafiat, J. Vepa, and M. Lincoln. 2007. The AMI system for transcription of speech in meetings. In *Proc. of ICASSP 2007,*, pages 357–360.

B. Klimt and Y. Yang. 2004. Introducing the enron corpus. In *Proc. of CEAS 2004, Mountain View, CA, USA*.

S. Maskey and J. Hirschberg. 2005. Comparing lexial, acoustic/prosodic, discourse and structural features for speech summarization. In *Proc. of Interspeech 2005, Lisbon, Portugal*, pages 621–624.

S. Muresan, E. Tzoukermann, and J. Klavans. 2001. Combining linguistic and machine learning techniques for email summarization. In *Proc. of ConLL 2001, Toulouse, France*.

G. Murray and S. Renals. 2008. Meta comments for summarizing meeting speech. In *Proc. of MLMI 2008, Utrecht, Netherlands*.

G. Murray, S. Renals, and J. Carletta. 2005a. Extractive summarization of meeting recordings. In *Proc. of Interspeech 2005, Lisbon, Portugal*, pages 593–596.

G. Murray, S. Renals, J. Carletta, and J. Moore. 2005b. Evaluating automatic summaries of meeting recordings. In *Proc. of the ACL 2005 MTSE Workshop, Ann Arbor, MI, USA*, pages 33–40.

G. Murray, S. Renals, J. Moore, and J. Carletta. 2006. Incorporating speaker and discourse features into speech summarization. In *Proc. of the HLT-NAACL 2006, New York City, USA*, pages 367–374.

G. Murray. 2007. *Using Speech-Specific Features for Automatic Speech Summarization*. Ph.D. thesis, University of Edinburgh.

A. Nenkova and B. Passonneau. 2004. Evaluating content selection in summarization: The Pyramid method. In *Proc. of HLT-NAACL 2004, Boston, MA, USA*, pages 145–152.

O. Rambow, L. Shrestha, J. Chen, and C. Lauridsen. 2004. Summarizing email threads. In *Proc. of HLT-NAACL 2004, Boston, USA*.

R. Valenza, T. Robinson, M. Hickey, and R. Tucker. 1999. Summarization of spoken audio through information extraction. In *Proc. of the ESCA Workshop on Accessing Information in Spoken Audio, Cambridge UK*, pages 111–116.

A. Waibel, M. Bett, M. Finke, and R. Stiefelhagen. 1998. Meeting browser: Tracking and summarizing meetings. In D. E. M. Penrose, editor, *Proc. of the Broadcast News Transcription and Understanding Workshop, Lansdowne, VA, USA*, pages 281–286.

J. Yeh and A. Harnly. 2006. Email thread reassembly using similarity matching. In *Proc of CEAS 2006*.

D. Zajic, B. Dorr, and J. Lin. 2008. Single-document and multi-document summarization techniques for email threads using sentence compression. *Information Processing and Management, to appear*.

K. Zechner. 2002. Automatic summarization of open-domain multiparty dialogues in diverse genres. *Computational Linguistics*, 28(4):447–485.

A Generative Model for Parsing Natural Language to Meaning Representations

Wei Lu[1], **Hwee Tou Ng**[1,2], **Wee Sun Lee**[1,2]
[1]Singapore-MIT Alliance
[2]Department of Computer Science
National University of Singapore
`luwei@nus.edu.sg`
`{nght,leews}@comp.nus.edu.sg`

Luke S. Zettlemoyer
CSAIL
Massachusetts Institute of Technology
`lsz@csail.mit.edu`

Abstract

In this paper, we present an algorithm for learning a generative model of natural language sentences together with their formal meaning representations with hierarchical structures. The model is applied to the task of mapping sentences to hierarchical representations of their underlying meaning. We introduce dynamic programming techniques for efficient training and decoding. In experiments, we demonstrate that the model, when coupled with a discriminative reranking technique, achieves state-of-the-art performance when tested on two publicly available corpora. The generative model degrades robustly when presented with instances that are different from those seen in training. This allows a notable improvement in recall compared to previous models.

1 Introduction

To enable computers to understand natural human language is one of the classic goals of research in natural language processing. Recently, researchers have developed techniques for learning to map sentences to hierarchical representations of their underlying meaning (Wong and Mooney, 2006; Kate and Mooney, 2006).

One common approach is to learn some form of probabilistic grammar which includes a list of lexical items that models the meanings of input words and also includes rules for combining lexical meanings to analyze complete sentences. This approach performs well but is constrained by the use of a single, learned grammar that contains a fixed set of lexical entries and productions. In practice, such a grammar may lack the rules required to correctly parse some of the new test examples.

In this paper, we develop an alternative approach that learns a model which does not make use of an explicit grammar but, instead, models the correspondence between sentences and their meanings with a generative process. This model is defined over *hybrid trees* whose nodes include both natural language words and meaning representation tokens. Inspired by the work of Collins (2003), the generative model builds trees by recursively creating nodes at each level according to a Markov process. This implicit grammar representation leads to flexible learned models that generalize well. In practice, we observe that it can correctly parse a wider range of test examples than previous approaches.

The generative model is learned from data that consists of sentences paired with their meaning representations. However, there is no explicit labeling of the correspondence between words and meaning tokens that is necessary for building the hybrid trees. This creates a challenging, hidden-variable learning problem that we address with the use of an inside-outside algorithm. Specifically, we develop a dynamic programming parsing algorithm that leads to $O(n^3m)$ time complexity for inference, where n is the sentence length and m is the size of meaning structure. This approach allows for efficient training and decoding.

In practice, we observe that the learned generative models are able to assign a high score to the correct meaning for input sentences, but that this correct meaning is not always the highest scoring option.

To address this problem, we use a simple reranking approach to select a parse from a k-best list of parses. This pipelined approach achieves state-of-the-art performance on two publicly available corpora. In particular, the flexible generative model leads to notable improvements in recall, the total percentage of sentences that are correctly parsed.

2 Related Work

In Section 9, we will compare performance with the three existing systems that were evaluated on the same data sets we consider. SILT (Kate et al., 2005) learns deterministic rules to transform either sentences or their syntactic parse trees to meaning structures. WASP (Wong and Mooney, 2006) is a system motivated by statistical machine translation techniques. It acquires a set of synchronous lexical entries by running the IBM alignment model (Brown et al., 1993) and learns a log-linear model to weight parses. KRISP (Kate and Mooney, 2006) is a discriminative approach where meaning representation structures are constructed from the natural language strings hierarchically. It is built on top of SVM^{struct} with string kernels.

Additionally, there is substantial related research that is not directly comparable to our approach. Some of this work requires different levels of supervision, including labeled syntactic parse trees (Ge and Mooney, 2005; Ge and Mooney, 2006). Others do not perform lexical learning (Tang and Mooney, 2001). Finally, recent work has explored learning to map sentences to lambda-calculus meaning representations (Wong and Mooney, 2007; Zettlemoyer and Collins, 2005; Zettlemoyer and Collins, 2007).

3 Meaning Representation

We restrict our meaning representation (MR) formalism to a variable free version as presented in (Wong and Mooney, 2006; Kate et al., 2005).

A training instance consists of a natural language sentence (NL sentence) and its corresponding meaning representation structure (MR structure). Consider the following instance taken from the GEOQUERY corpus (Kate et al., 2005):

The NL sentence "`How many states do not have rivers ?`" consists of 8 words, including punctuation. The MR is a hierarchical tree structure, as shown in Figure 1.

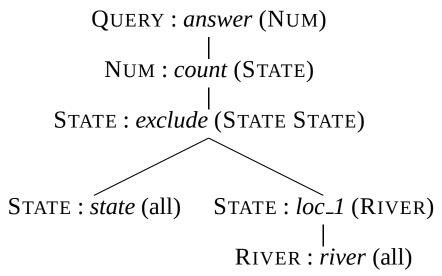


Figure 1: An example MR structure

Following an inorder traversal of this MR tree, we can equivalently represent it with the following list of *meaning representation productions* (MR productions):

(0)	QUERY	:	*answer* (NUM)
(1)	NUM	:	*count* (STATE)
(2)	STATE	:	*exclude* (STATE$_1$ STATE$_2$)
(3)	STATE	:	*state* (all)
(4)	STATE	:	*loc_1* (RIVER)
(5)	RIVER	:	*river* (all)

Each such MR production consists of three components: a *semantic category*, a *function symbol* which can be omitted (considered empty), and a list of *arguments*. An argument can be either a child semantic category or a constant. Take production (1) for example: it has a semantic category "NUM", a function symbol "*count*", and a child semantic category "STATE" as its only argument. Production (5) has "RIVER" as its semantic category, "*river*" as the function symbol, and "all" is a constant.

4 The Generative Model

We describe in this section our proposed generative model, which simultaneously generates a NL sentence and an MR structure.

We denote a single NL word as w, a contiguous sequence of NL words as $\mathbf{w}$, and a complete NL sentence as $\widehat{\mathbf{w}}$. In the MR structure, we denote a semantic category as $\mathcal{M}$. We denote a single MR production as m_a, or $\mathcal{M}_a : p_\alpha(\mathcal{M}_b, \mathcal{M}_c)$, where $\mathcal{M}_a$ is the semantic category for this production, p_α is the function symbol, and $\mathcal{M}_b, \mathcal{M}_c$ are the child semantic categories. We denote $\mathbf{m}_a$ as an MR structure rooted by an MR production m_a, and $\widehat{\mathbf{m}_a}$ an MR structure for a complete sentence rooted by an MR production m_a.

The model generates a *hybrid tree* that represents a sentence $\widehat{\mathbf{w}} = \mathbf{w}_1 \ldots \mathbf{w}_2 \ldots$ paired with an MR structure $\widehat{\mathbf{m}_a}$ rooted by m_a.

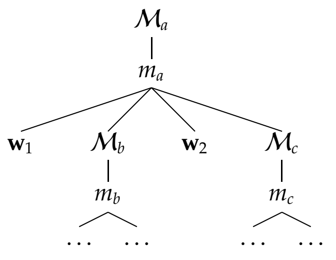


Figure 2: The generation process

Figure 2 shows part of a hybrid tree that is generated as follows. Given a semantic category $\mathcal{M}_a$, we first pick an MR production m_a that has the form $\mathcal{M}_a : p_\alpha(\mathcal{M}_b, \mathcal{M}_c)$, which gives us the function symbol p_α as well as the child semantic categories $\mathcal{M}_b$ and $\mathcal{M}_c$. Next, we generate the *hybrid sequence* of child nodes $\overline{\mathbf{w}_1\ \mathcal{M}_b\ \mathbf{w}_2\ \mathcal{M}_c}$, which consists of NL words and semantic categories.

After that, two child MR productions m_b and m_c are generated. These two productions will in turn generate other hybrid sequences and productions, recursively. This process produces a hybrid tree $\mathcal{T}$, whose nodes are either NL words or MR productions. Given this tree, we can recover a NL sentence $\mathbf{w}$ by recording the NL words visited in depth-first traversal order and can recover an MR structure $\mathbf{m}$ by following a tree-specific traversal order, defined by the hybrid-patterns we introduce below. Figure 3 gives a partial hybrid tree for the training example from Section 3. Note that the leaves of a hybrid tree are always NL tokens.

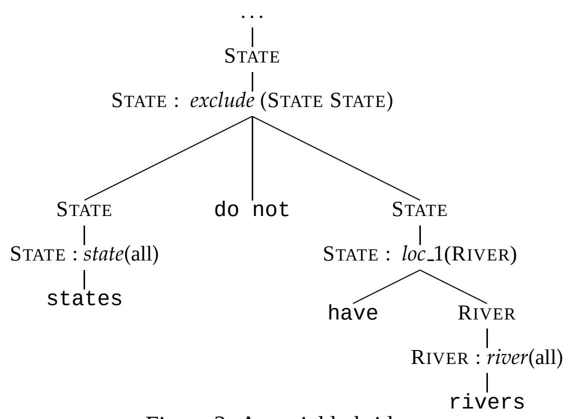


Figure 3: A partial hybrid tree

With several independence assumptions, the probability of generating $\langle\widehat{\mathbf{w}}, \widehat{\mathbf{m}}, \mathcal{T}\rangle$ is defined as:

$$\begin{aligned} P(\widehat{\mathbf{w}}, \widehat{\mathbf{m}}, \mathcal{T}) = P(\mathcal{M}_a) \times P(m_a|\mathcal{M}_a) \times P(\overline{\mathbf{w}_1\ \mathcal{M}_b\ \mathbf{w}_2\ \mathcal{M}_c}|m_a) \\ \times P(m_b|m_a, \text{arg} = 1) \times P(\ldots|m_b) \\ \times P(m_c|m_a, \text{arg} = 2) \times P(\ldots|m_c) \end{aligned} \quad (1)$$

where "arg" refers to the position of the child semantic category in the argument list.

Motivated by Collins' syntactic parsing models (Collins, 2003), we consider the generation process for a hybrid sequence from an MR production as a Markov process.

Given the assumption that each MR production has at most two semantic categories in its arguments (any production can be transformed into a sequence of productions of this form), Table 1 includes the list of all possible *hybrid patterns*.

# RHS	Hybrid Pattern	# Patterns
0	$m \to \mathbf{w}$	1
1	$m \to [\mathbf{w}]\mathcal{Y}[\mathbf{w}]$	4
2	$m \to [\mathbf{w}]\mathcal{Y}[\mathbf{w}]\mathcal{Z}[\mathbf{w}]$	8
	$m \to [\mathbf{w}]\mathcal{Z}[\mathbf{w}]\mathcal{Y}[\mathbf{w}]$	8

Table 1: A list of hybrid patterns, [] denotes optional

In this table, m is an MR production, $\mathcal{Y}$ and $\mathcal{Z}$ are respectively the first and second child semantic category in m's argument list. The symbol $\mathbf{w}$ refers to a contiguous sequence of NL words, and anything inside [] can be optionally omitted. The last row contains hybrid patterns that reflect reordering of one production's child semantic categories during the generation process. For example, consider the case that the MR production STATE : *exclude* (STATE$_1$ STATE$_2$) generates a hybrid sequence $\overline{\text{STATE}_1\ \texttt{do not}\ \text{STATE}_2}$, the hybrid pattern $m \to \mathcal{Y}\mathbf{w}\mathcal{Z}$ is associated with this generation step.

For the example hybrid tree in Figure 2, we can decompose the probability for generating the hybrid sequence as follows:

$$\begin{aligned} P(\overline{\mathbf{w}_1\ \mathcal{M}_b\ \mathbf{w}_2\ \mathcal{M}_c}|m_a) = P(m \to \mathbf{w}\mathcal{Y}\mathbf{w}\mathcal{Z}|m_a) \times P(\mathbf{w}_1|m_a) \\ \times P(\mathcal{M}_b|m_a, \mathbf{w}_1) \times P(\mathbf{w}_2|m_a, \mathbf{w}_1, \mathcal{M}_b) \\ \times P(\mathcal{M}_c|m_a, \mathbf{w}_1, \mathcal{M}_b, \mathbf{w}_2) \times P(\text{END}|m_a, \mathbf{w}_1, \mathcal{M}_b, \mathbf{w}_2, \mathcal{M}_c) \end{aligned} \quad (2)$$

Note that unigram, bigram, or trigram assumptions can be made here for generating NL words and semantic categories. For example, under a bigram assumption, the second to last term can be written as $P(\mathcal{M}_c|m_a, \mathbf{w}_1, \mathcal{M}_b, \mathbf{w}_2) \equiv P(\mathcal{M}_c|m_a, w_2^k)$, where w_2^k is the last word in $\mathbf{w}_2$. We call such additional information that we condition on, the *context*.

Note that our generative model is different from the synchronous context free grammars (SCFG) in a number of ways. A standard SCFG produces a correspondence between a pair of trees while our model produces a single hybrid tree that represents the correspondence between a sentence and a tree. Also, SCFGs use a finite set of context-free rewrite rules to define the model, where the rules are possibly weighted. In contrast, we make use of the more flexible Markov models at each level of the generative process, which allows us to potentially produce a far wider range of possible trees.

5 Parameter Estimation

There are three categories of parameters used in the model. The first category of parameters models the generation of new MR productions from their parent MR productions: *e.g.*, $P(m_b|m_a, \text{arg} = 1)$; the second models the generation of a hybrid sequence from an MR production: *e.g.*, $P(\mathbf{w}_1|m_a)$, $P(\mathcal{M}_b|m_a, \mathbf{w}_1)$; the last models the selection of a hybrid pattern given an MR production, *e.g.*, $P(m \to \mathbf{w}\mathcal{Y}|m_a)$. We will estimate parameters from all categories, with the following constraints:

1. $\sum_{m'} \rho(m'|m_j, \text{arg}{=}k){=}1$ for all j and $k = 1, 2$.

   These parameters model the MR structures, and can be referred to as *MR model parameters*.

2. $\sum_t \theta(t|m_j, \Lambda){=}1$ for all j, where t is a NL word, the "END" symbol, or a semantic category. Λ is the context associated with m_j and t.

   These parameters model the emission of NL words, the "END" symbol, and child semantic categories from an MR production. We call them *emission parameters*.

3. $\sum_r \phi(r|m_j) = 1$ for all j, where r is a hybrid pattern listed in Table 1.

   These parameters model the selection of hybrid patterns. We name them *pattern parameters*.

With different context assumptions, we reach different variations of the model. In particular, we consider three assumptions, as follows:

Model I We make the following assumption:

$$\theta(t_k|m_j, \Lambda) = P(t_k|m_j) \tag{3}$$

where t_k is a semantic category or a NL word, and m_j is an MR production.

In other words, generation of the next NL word depends on its direct parent MR production only. Such a *Unigram Model* may help in recall (the number of correct outputs over the total number of inputs), because it requires the least data to estimate.

Model II We make the following assumption:

$$\theta(t_k|m_j, \Lambda) = P(t_k|m_j, t_{k-1}) \tag{4}$$

where t_{k-1} is the semantic category or NL word to the left of t_k, *i.e.*, the previous semantic category or NL word.

In other words, generation of the next NL word depends on its direct parent MR production as well as the previously generated NL word or semantic category only. This model is also referred to as *Bigram Model*. This model may help in precision (the number of correct outputs over the total number of outputs), because it conditions on a larger context.

Model III We make the following assumption:

$$\theta(t_k|m_j, \Lambda) = \frac{1}{2} \times \Big(P(t_k|m_j) + P(t_k|m_j, t_{k-1})\Big) \tag{5}$$

We can view this model, called the *Mixgram Model*, as an interpolation between Model I and II. This model gives us a balanced score for both precision and recall.

5.1 Modeling Meaning Representation

The MR model parameters can be estimated independently from the other two. These parameters can be viewed as the "language model" parameters for the MR structure, and can be estimated directly from the corpus by simply reading off the counts of occurrences of MR productions in MR structures over the training corpus. To resolve data sparseness problem, a variant of the bigram Katz Back-Off Model (Katz, 1987) is employed here for smoothing.

5.2 Learning the Generative Parameters

Learning the remaining two categories of parameters is more challenging. In a conventional PCFG parsing task, during the training phase, the correct correspondence between NL words and syntactic structures is fully accessible. In other words, there is a single deterministic derivation associated with each training instance. Therefore model parameters can be directly estimated from the training corpus by counting. However, in our task, the correct correspondence between NL words and MR structures is unknown. Many possible derivations could reach the same NL-MR pair, where each such derivation forms a hybrid tree.

The hybrid tree is constructed using hidden variables and estimated from the training set. An efficient inside-outside style algorithm can be used for model estimation, similar to that used in (Yamada and Knight, 2001), as discussed next.

5.2.1 The Inside-Outside Algorithm with EM

In this section, we discuss how to estimate the emission and pattern parameters with the Expectation Maximization (EM) algorithm (Dempster et al., 1977), by using an inside-outside (Baker, 1979) dynamic programming approach.

Denote $\mathbf{n}^i \equiv \langle \mathbf{m}^i, \mathbf{w}^i \rangle$ as the i-th training instance, where $\mathbf{m}^i$ and $\mathbf{w}^i$ are the MR structure and the NL sentence of the i-th instance respectively. We also denote $\mathbf{n}_v \equiv \langle \mathbf{m}_v, \mathbf{w}_v \rangle$ as an aligned pair of MR substructure and contiguous NL substring, where the MR substructure rooted by MR production m_v will correspond to (*i.e.*, hierarchically generate) the NL substring $\mathbf{w}_v$. The symbol h is used to denote a hybrid sequence, and the function *Parent*(h) gives the unique MR substructure-NL subsequence pair which can be decomposed as h. *Parent*($\mathbf{n}_v$) returns the set of all possible hybrid sequences under which the pair $\mathbf{n}_v$ can be generated. Similarly, *Children*(h) gives the NL-MR pairs that appear directly below the hybrid sequence h in a hybrid tree, and *Children*($\mathbf{n}$) returns the set of all possible hybrid sequences that $\mathbf{n}$ can be decomposed as. Figure 4 gives a packed tree structure representing the relations between the entities.

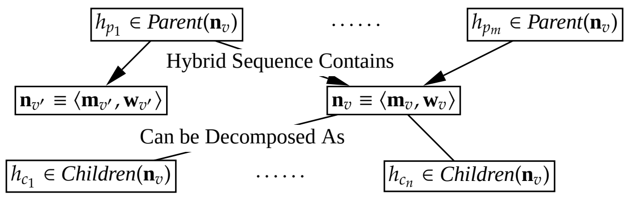

$h_{p_1} \in Parent(\mathbf{n}_v)$
......
$h_{p_m} \in Parent(\mathbf{n}_v)$
Hybrid Sequence Contains
$\mathbf{n}_{v'} \equiv \langle \mathbf{m}_{v'}, \mathbf{w}_{v'} \rangle$
$\mathbf{n}_v \equiv \langle \mathbf{m}_v, \mathbf{w}_v \rangle$
Can be Decomposed As
$h_{c_1} \in Children(\mathbf{n}_v)$
......
$h_{c_n} \in Children(\mathbf{n}_v)$

Figure 4: A packed tree structure representing the relations between hybrid sequences and NL-MR pairs

The formulas for computing inside and outside probabilities as well as the equations for updating parameters are given in Figure 5. We use a CKY-style parse chart for tracking the probabilities.

5.2.2 Smoothing

It is reasonable to believe that different MR productions that share identical function symbols are likely to generate NL words with similar distribution, regardless of semantic categories. For example,

The inside (β) probabilities are defined as

- If $\mathbf{n}_v \equiv \langle \mathbf{m}_v, \mathbf{w}_v \rangle$ is leaf
$$\beta(\mathbf{n}_v) = P(\mathbf{w}_v|\mathbf{m}_v) \tag{6}$$
- If $\mathbf{n}_v \equiv \langle \mathbf{m}_v, \mathbf{w}_v \rangle$ is not leaf
$$\beta(\mathbf{n}_v) = \sum_{h \in Children(\mathbf{n}_v)} \Big(P(h|\mathbf{m}_v) \times \prod_{\mathbf{n}_{v'} \in Children(h)} \beta(\mathbf{n}_{v'}) \Big) \tag{7}$$

The outside (α) probabilities are defined as

- If $\mathbf{n}_v \equiv \langle \mathbf{m}_v, \mathbf{w}_v \rangle$ is root
$$\alpha(\mathbf{n}_v) = 1 \tag{8}$$
- If $\mathbf{n}_v \equiv \langle \mathbf{m}_v, \mathbf{w}_v \rangle$ is not root
$$\alpha(\mathbf{n}_v) = \sum_{h \in Parent(\mathbf{n}_v)} \Big(\alpha\big(Parent(h)\big) \times P\big(h|Parent(h)\big) \times \prod_{\mathbf{n}_{v'} \in Children(h), v' \neq v} \beta(\mathbf{n}_{v'}) \Big) \tag{9}$$

Parameter Update

- Update the emission parameter
The count $c^i(t, m_v, \Lambda_k)$, where t is a NL word or a semantic category, for an instance pair $\mathbf{n}^i \equiv \langle \mathbf{m}^i, \mathbf{w}^i \rangle$:
$$c^i(t, m_v, \Lambda_k) = \frac{1}{\beta(\mathbf{n}^i)} \times \sum_{(t, m_v, \Lambda_k) \text{ in } h \in Children(m_v)} \Big(\alpha(\mathbf{n}^i_v) \times P(h|m_v) \times \prod_{\mathbf{n}^i_{v'} \in Children(h)} \beta(\mathbf{n}^i_{v'}) \Big)$$
The emission parameter is re-estimated as:
$$\theta'(t|m_v, \Lambda_k) = \frac{\sum_i c^i(t, m_v, \Lambda_k)}{\sum_{t'} \sum_i c^i(t', m_v, \Lambda_k)} \tag{10}$$
- Update the pattern parameter
The count $c^i(r, m_v)$, where r is a hybrid pattern, for an instance pair $\mathbf{n}^i \equiv \langle \mathbf{m}^i, \mathbf{w}^i \rangle$:
$$c^i(r, m_v) = \frac{1}{\beta(\mathbf{n}^i)} \times \sum_{(r, m_v) \text{ in } h \in Children(m_v)} \Big(\alpha(\mathbf{n}^i_v) \times P(h|m_v) \times \prod_{\mathbf{n}^i_{v'} \in Children(h)} \beta(\mathbf{n}^i_{v'}) \Big)$$
The pattern parameter is re-estimated as:
$$\phi'(r|m_v) = \frac{\sum_i c^i(r, m_v)}{\sum_{r'} \sum_i c^i(r', m_v)} \tag{11}$$

Figure 5: The inside/outside formulas as well as update equations for EM

RIVER : *largest* (RIVER) and CITY : *largest* (CITY) are both likely to generate the word "`biggest`".

In view of this, a smoothing technique is deployed. We assume half of the time words can be generated from the production's function symbol alone if it is not empty. Mathematically, assuming m_a with function symbol p_a, for a NL word or semantic category t, we have:

$$\theta(t|m_a,\Lambda) = \begin{cases} \theta_e(t|m_a,\Lambda) & \text{If } p_a \text{ is empty} \\ \big(\theta_e(t|m_a,\Lambda) + \theta_e(t|p_a,\Lambda)\big)/2 & \text{otherwise} \end{cases}$$

where θ_e models the generation of t from an MR production or its function symbol, together with the context Λ.

6 A Dynamic Programming Algorithm for Inside-Outside Computation

Though the inside-outside approach already employs packed representations for dynamic programming, a naive implementation of the inference algorithm will still require $O(n^6m)$ time for 1 EM iteration, where n and m are the length of the NL sentence and the size of the MR structure respectively. This is not very practical as in one of the corpora we look at, n and m can be up to 45 and 20 respectively.

In this section, we develop an efficient dynamic programming algorithm that enables the inference to run in $O(n^3m)$ time. The idea is as follows. Instead of treating each possible hybrid sequence as a separate rule, we efficiently aggregate the already computed probability scores for hybrid sequences that share identical hybrid patterns. Such aggregated scores can then be used for subsequent computations. By doing this, we can effectively avoid a large amount of redundant computations. The algorithm supports both unigram and bigram context assumptions. For clarity and ease of presentation, we primarily make the unigram assumption throughout our discussion.

We use $\beta(\mathbf{m}_v, \mathbf{w}_v)$ to denote the inside probability for $\mathbf{m}_v$-$\mathbf{w}_v$ pair, $b_r[\mathbf{m}_v, \mathbf{w}_v, c]$ to denote the aggregated probabilities for the MR sub-structure $\mathbf{m}_v$ to generate all possible hybrid sequences based on $\mathbf{w}_v$ with pattern r that covers its c-th child only. In addition, we use $\mathbf{w}_{(i,j)}$ to denote a subsequence of $\mathbf{w}$ with start index i (inclusive) and end index j (exclusive). We also use $\beta_r[\![\mathbf{m}_v, \mathbf{w}_v]\!]$ to denote the aggregated inside probability for the pair $\langle \mathbf{m}_v, \mathbf{w}_v \rangle$, if the hybrid pattern is restricted to r only. By definition we have:

$$\beta(\mathbf{m}_v, \mathbf{w}_v) = \sum_r \phi(r|m_v) \times \beta_r[\![\mathbf{m}_v, \mathbf{w}_v]\!] \times \theta(\mathsf{END}|m_v) \quad (12)$$

Relations between β_r and b_r can also be established. For example, if m_v has one child semantic category, we have:

$$\beta_{m\rightarrow \mathbf{w}\mathcal{Y}}[\![\mathbf{m}_v, \mathbf{w}_v]\!] = b_{m\rightarrow \mathbf{w}\mathcal{Y}}[\mathbf{m}_v, \mathbf{w}_v, 1] \quad (13)$$

For the case when m_v has two child semantic categories as arguments, we have, for example:

$$\beta_{m\rightarrow \mathbf{w}\mathcal{Y}\mathcal{Z}\mathbf{w}}[\![\mathbf{m}_v, \mathbf{w}_{(i,j)}]\!] = \sum_{i+2\le k\le j-2} b_{m\rightarrow \mathbf{w}\mathcal{Y}}[m_v, \mathbf{w}_{(i,k)}, 1] \times b_{m\rightarrow \mathcal{Y}\mathbf{w}}[m_v, \mathbf{w}_{(k,j)}, 2] \quad (14)$$

Note that there also exist relations amongst b terms for more efficient computation, for example:

$$b_{m\rightarrow \mathbf{w}\mathcal{Y}}[\mathbf{m}_v, \mathbf{w}_{(i,j)}, c] = \theta(w_i|m_v) \times \Big(b_{m\rightarrow \mathbf{w}\mathcal{Y}}[\mathbf{m}_v, \mathbf{w}_{(i+1,j)}, c] + b_{m\rightarrow \mathcal{Y}}[\mathbf{m}_v, \mathbf{w}_{(i+1,j)}, c]\Big) \quad (15)$$

Analogous but more complex formulas are used for computing the outside probabilities. Updating of parameters can be incorporated into the computation of outside probabilities efficiently.

7 Decoding

In the decoding phase, we want to find the optimal MR structure $\widehat{\mathbf{m}}^*$ given a new NL sentence $\widehat{\mathbf{w}}$:

$$\widehat{\mathbf{m}}^* = \arg\max_{\widehat{\mathbf{m}}} P(\widehat{\mathbf{m}}|\widehat{\mathbf{w}}) = \arg\max_{\widehat{\mathbf{m}}} \sum_{\mathcal{T}} P(\widehat{\mathbf{m}}, \mathcal{T}|\widehat{\mathbf{w}}) \quad (16)$$

where $\mathcal{T}$ is a possible hybrid tree associated with the $\widehat{\mathbf{m}}$-$\widehat{\mathbf{w}}$ pair. However, it is expensive to compute the summation over all possible hybrid trees. We therefore find the most likely hybrid tree instead:

$$\widehat{\mathbf{m}}^* = \arg\max_{\widehat{\mathbf{m}}} \max_{\mathcal{T}} P(\widehat{\mathbf{m}}, \mathcal{T}|\widehat{\mathbf{w}}) = \arg\max_{\widehat{\mathbf{m}}} \max_{\mathcal{T}} P(\widehat{\mathbf{w}}, \widehat{\mathbf{m}}, \mathcal{T}) \quad (17)$$

We have implemented an exact top-k decoding algorithm for this task. Dynamic programming techniques similar to those discussed in Section 6 can also be applied when retrieving the top candidates.

We also find the Viterbi hybrid tree given a NL-MR pair, which can be done in an analogous way. This tree will be useful for reranking.

8 Reranking and Filtering of Predictions

Due to the various independence assumptions we have made, the model lacks the ability to express some long range dependencies. We therefore post-process the best candidate predictions with a discriminative reranking algorithm.

Feature Type	Description	Example
1. Hybrid Rule	A MR production and its child hybrid form	f_1 : STATE : *loc_1*(RIVER) → `have` RIVER
2. Expanded Hybrid Rule	A MR production and its child hybrid form expanded	f_2 : STATE : *loc_1*(RIVER) → ⟨`have`, RIVER : *river*(all)⟩
3. Long-range Unigram	A MR production and a NL word appearing below in tree	f_3 : STATE : *exclude*(STATE STATE) → `rivers`
4. Grandchild Unigram	A MR production and its grandchild NL word	f_4 : STATE : *loc_1*(RIVER) → `rivers`
5. Two Level Unigram	A MR production, its parent production, and its child NL word	f_5 : ⟨RIVER : *river*(all), STATE : *loc_1*(RIVER)⟩ → `rivers`
6. Model Log-Probability	Logarithm of base model's joint probability	$\log\left(\widehat{P}(\mathbf{w}, \mathbf{m}, \mathcal{T})\right)$.

Table 2: All the features used. There is one feature for each possible combination, under feature type 1-5. It takes value 1 if the combination is present, and 0 otherwise. Feature 6 takes real values.

8.1 The Averaged Perceptron Algorithm with Separating Plane

The averaged perceptron algorithm (Collins, 2002) has previously been applied to various NLP tasks (Collins, 2002; Collins, 2001) for discriminative reranking. The detailed algorithm can be found in (Collins, 2002). In this section, we extend the conventional averaged perceptron by introducing an explicit separating plane on the feature space.

Our reranking approach requires three components during training: a **GEN** function that defines for each NL sentence a set of candidate hybrid trees; a single correct reference hybrid tree for each training instance; and a feature function Φ that defines a mapping from a hybrid tree to a feature vector. The algorithm learns a weight vector $\mathbf{w}$ that associates a weight to each feature, such that a score $\mathbf{w}\cdot\Phi(\mathcal{T})$ can be assigned to each candidate hybrid tree $\mathcal{T}$. Given a new instance, the hybrid tree with the highest score is then picked by the algorithm as the output.

In this task, the **GEN** function is defined as the output hybrid trees of the top-k (k is set to 50 in our experiments) decoding algorithm, given the learned model parameters. The correct reference hybrid tree is determined by running the Viterbi algorithm on each training NL-MR pair. The feature function is discussed in section 8.2.

While conventional perceptron algorithms usually optimize the accuracy measure, we extend it to allow optimization of the F-measure by introducing an explicit separating plane on the feature space that rejects certain predictions even when they score highest. The idea is to find a threshold b after $\mathbf{w}$ is learned, such that a prediction with score below b gets rejected. We pick the threshold that leads to the optimal F-measure when applied to the training set.

8.2 Features

We list in Table 2 the set of features we used. Examples are given based on the hybrid tree in Figure 3. Some of the them are adapted from (Collins and Koo, 2005) for a natural language parsing task. Features 1-5 are indicator functions (*i.e.,* it takes value 1 if a certain combination as the ones listed in Table 2 is present, 0 otherwise), while feature 6 is real valued. Features that do not appear more than once in the training set are discarded.

9 Evaluation

Our evaluations were performed on two corpora, GEOQUERY and ROBOCUP. The GEOQUERY corpus contains MR defined by a Prolog-based language used in querying a database on U.S. geography. The ROBOCUP corpus contains MR defined by a coaching language used in a robot coaching competition. There are in total 880 and 300 instances for the two corpora respectively. Standard 10-fold cross validations were performed and the micro-averaged results are presented in this section. To make our system directly comparable to previous systems, all our experiments were based on identical training and test data splits of both corpora as reported in the experiments of Wong and Mooney (2006).

9.1 Training Methodology

Given a training set, we first run a variant of IBM alignment model 1 (Brown et al., 1993) for 100 iterations, and then initialize Model I with the learned parameter values. This IBM model is a word-to-word alignment model that does not model word order, so we do not have to linearize the hierarchical MR structure. Given this initialization, we train Model I for 100 EM iterations and use the learned parameters to initialize Model II which is trained for another 100 EM iterations. Model III is simply an interpolation of the above two models. As for the reranking phase, we initialize the weight vector with the zero vector $\mathbf{0}$, and run the averaged perceptron algorithm for 10 iterations.

9.2 Evaluation Methodology

Following Wong (2007) and other previous work, we report performance in terms of *Precision* (percentage of answered NL sentences that are correct), *Recall* (percentage of correctly answered NL sentences, out of all NL sentences) and *F*-score (harmonic mean of *Precision* and *Recall*).

Again following Wong (2007), we define the correct output MR structure as follows. For the GEOQUERY corpus, an MR structure is considered correct if and only if it retrieves identical results as the reference MR structure when both are issued as queries to the underlying Prolog database. For the ROBOCUP corpus, an MR structure is considered correct if and only if it has the same string representation as the reference MR structure, up to reordering of children of MR productions whose function symbols are commutative, such as *and*, *or*, etc.

9.3 Comparison over Three Models

Model	GEOQUERY (880)			ROBOCUP (300)		
	Prec.	*Rec.*	*F*	*Prec.*	*Rec.*	*F*
I	81.3	77.1	79.1	71.1	**64.0**	67.4
II	**89.0**	76.0	82.0	**82.4**	57.7	**67.8**
III	86.2	**81.8**	**84.0**	70.4	63.3	66.7
I+R	87.5	80.5	83.8	79.1	67.0	72.6
II+R	**93.2**	73.6	82.3	**88.4**	56.0	68.6
III+R	89.3	**81.5**	**85.2**	82.5	**67.7**	**74.4**

Table 3: Performance comparison over three models (*Prec.*:precision, *Rec.*:recall, +R: with reranking)

We evaluated the three models, with and without reranking. The results are presented in Table 3. Comparing Model I and Model II, we noticed that for both corpora, Model I in general achieves better recall while Model II achieves better precision. This observation conforms to our earlier expectations. Model III, as an interpolation of the above two models, achieves a much better *F*-measure on GEOQUERY corpus. However, it is shown to be less effective on ROBOCUP corpus. We noticed that compared to the GEOQUERY corpus, ROBOCUP corpus contains longer sentences, larger MR structures, and a significant amount of non-compositionality. These factors combine to present a challenging problem for parsing with the generative model. Interestingly, although Model III fails to produce better best predictions for this corpus, we found that its top-*k* list contains a relatively larger number of correct predictions than Model I or Model II. This indicates the possibility of enhancing the performance with reranking.

The reranking approach is shown to be quite effective. We observe a consistent improvement in both precision and *F*-measure after employing the reranking phase for each model.

9.4 Comparison with Other Models

Among all the previous models, SILT, WASP, and KRISP are directly comparable to our model. They required the same amount of supervision as our system and were evaluated on the same corpora.

We compare our model with these models in Table 4, where the performance scores for the previous systems are taken from (Wong, 2007). For GEOQUERY corpus, our model performs substantially better than all the three previous models, with a notable improvement in the recall score. In fact, if we look at the recall scores alone, our best-performing model achieves a 6.7% and 9.8% absolute improvement over two other state-of-the-art models WASP and KRISP respectively. This indicates that overall, our model is able to handle over 25% of the inputs that could not be handled by previous systems. On the other hand, in terms of *F*-measure, we gain a 4.1% absolute improvement over KRISP, which leads to an error reduction rate of 22%. On the ROBOCUP corpus, our model's performance is also ranked the highest[1].

System	GEOQUERY (880)			ROBOCUP (300)		
	Prec.	*Rec.*	*F*	*Prec.*	*Rec.*	*F*
SILT	89.0	54.1	67.3	83.9	50.7	63.2
WASP	87.2	74.8	80.5	**88.9**	61.9	73.0
KRISP	**93.3**	71.7	81.1	85.2	61.9	71.7
Model III+R	89.3	**81.5**	**85.2**	82.5	**67.7**	**74.4**

Table 4: Performance comparison with other directly comparable systems

9.5 Performance on Other Languages

As a generic model that requires minimal assumptions on the natural language, our model is natural language independent and is able to handle various other natural languages than English. To validate this point, we evaluated our system on a subset of

[1]We are unable to perform statistical significance tests because the detailed performance for each fold of previously published research work is not available.

the GEOQUERY corpus consisting of 250 instances, with four different NL annotations.

As we can see from Table 5, our model is able to achieve performance comparable to WASP as reported by Wong (2007).

System	English			Spanish		
	Prec.	*Rec.*	*F*	*Prec.*	*Rec.*	*F*
WASP	95.42	70.00	80.76	91.99	72.40	81.03
Model III+R	91.46	72.80	**81.07**	95.19	79.20	**86.46**
System	**Japanese**			**Turkish**		
	Prec.	*Rec.*	*F*	*Prec.*	*Rec.*	*F*
WASP	91.98	74.40	**82.86**	96.96	62.40	75.93
Model III+R	87.56	76.00	81.37	93.82	66.80	**78.04**

Table 5: Performance on different natural languages for GEOQUERY-250 corpus

Our model is generic, which requires no domain-dependent knowledge and should be applicable to a wide range of different domains. Like all research in this area, the ultimate goal is to scale to more complex, open-domain language understanding problems. In future, we would like to create a larger corpus in another domain with multiple natural language annotations to further evaluate the scalability and portability of our approach.

10 Conclusions

We presented a new generative model that simultaneously produces both NL sentences and their corresponding MR structures. The model can be effectively applied to the task of transforming NL sentences to their MR structures. We also developed a new dynamic programming algorithm for efficient training and decoding. We demonstrated that this approach, augmented with a discriminative reranking technique, achieves state-of-the-art performance when tested on standard benchmark corpora.

In future, we would like to extend the current model to have a wider range of support of MR formalisms, such as the one with lambda-calculus support. We are also interested in investigating ways to apply the generative model to the inverse task: generation of a NL sentence that explains a given MR structure.

Acknowledgments

The authors would like to thank Leslie Pack Kaelbling for her valuable feedback and comments on this research. The authors would also like to thank the anonymous reviewers for their thoughtful comments on this paper. The research is partially supported by ARF grant R-252-000-240-112.

References

J. K. Baker. 1979. Trainable grammars for speech recognition. *Journal of the Acoustical Society of America*, 65:S132.

P. F. Brown, S. A. D. Pietra, V. J. D. Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

M. Collins and T. Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70.

M. Collins. 2001. Ranking algorithms for named-entity extraction: boosting and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pages 489–496.

M. Collins. 2002. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 1–8.

M. Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38.

R. Ge and R. J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL 2005)*, pages 9–16.

R. Ge and R. J. Mooney. 2006. Discriminative reranking for semantic parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL 2006)*, pages 263–270.

R. J. Kate and R. J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL 2006)*, pages 913–920.

R. J. Kate, Y. W. Wong, and R. J. Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI 2005)*, pages 1062–1068.

S. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3):400–401.

L. R. Tang and R. J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the 12th European Conference on Machine Learning (ECML 2001)*, pages 466–477.

Y. W. Wong and R. J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2006)*, pages 439–446.

Y. W. Wong and R. J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL 2007)*, pages 960–967.

Y. W. Wong. 2007. *Learning for Semantic Parsing and Natural Language Generation Using Statistical Machine Translation Techniques*. Ph.D. thesis, The University of Texas at Austin.

K. Yamada and K. Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 523–530.

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*.

L. S. Zettlemoyer and M. Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, pages 678–687.

Learning with Compositional Semantics as Structural Inference for Subsentential Sentiment Analysis

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

Abstract

Determining the polarity of a sentiment-bearing expression requires more than a simple bag-of-words approach. In particular, words or constituents within the expression can interact with each other to yield a particular overall polarity. In this paper, we view such subsentential interactions in light of *compositional semantics*, and present a novel learning-based approach that incorporates structural inference motivated by compositional semantics into the learning procedure. Our experiments show that (1) simple heuristics based on compositional semantics can perform better than learning-based methods that do not incorporate compositional semantics (accuracy of 89.7% vs. 89.1%), but (2) a method that integrates compositional semantics into learning performs better than all other alternatives (90.7%). We also find that "content-word negators", not widely employed in previous work, play an important role in determining expression-level polarity. Finally, in contrast to conventional wisdom, we find that expression-level classification accuracy uniformly *decreases* as additional, potentially disambiguating, context is considered.

1 Introduction

Determining the polarity of sentiment-bearing expressions at or below the sentence level requires more than a simple bag-of-words approach. One of the difficulties is that words or constituents within the expression can interact with each other to yield a particular overall polarity. To facilitate our discussion, consider the following examples:

1: [I did [*not*]$^{\neg}$ have any [*doubt*]$^{-}$ about it.]$^{+}$
2: [The report [*eliminated*]$^{\neg}$ my [*doubt*]$^{-}$.]$^{+}$
3: [They could [*not*]$^{\neg}$ [*eliminate*]$^{\neg}$ my [*doubt*]$^{-}$.]$^{-}$

In the first example, "doubt" in isolation carries a negative sentiment, but the overall polarity of the sentence is positive because there is a *negator* "not", which flips the polarity. In the second example, both "eliminated" and "doubt" carry negative sentiment in isolation, but the overall polarity of the sentence is positive because "eliminated" acts as a negator for its argument "doubt". In the last example, there are effectively two negators – "not" and "eliminated" – which reverse the polarity of "doubt" twice, resulting in the negative polarity for the overall sentence.

These examples demonstrate that words or constituents interact with each other to yield the expression-level polarity. And a system that simply takes the majority vote of the polarity of individual words will not work well on the above examples. Indeed, much of the previous learning-based research on this topic tries to incorporate salient interactions by encoding them as features. One approach includes features based on *contextual valence shifters*[1] (Polanyi and Zaenen, 2004), which are words that affect the polarity or intensity of sentiment over neighboring text spans (e.g., Kennedy and Inkpen (2005), Wilson et al. (2005), Shaikh et al. (2007)). Another approach encodes frequent subsentential patterns (e.g., McDonald et al. (2007)) as features; these might indirectly capture some of the subsentential interactions that affect polarity. How-

[1]For instance, "never", "nowhere", "little", "most", "lack", "scarcely", "deeply".

ever, both types of approach are based on learning models with a flat bag-of-features: some structural information can be encoded as higher order features, but the final representation of the input is still a flat feature vector that is inherently too limited to adequately reflect the complex structural nature of the underlying subsentential interactions. (Liang et al., 2008)

Moilanen and Pulman (2007), on the other hand, handle the structural nature of the interactions more directly using the ideas from *compositional semantics* (e.g., Montague (1974), Dowty et al. (1981)). In short, *the Principle of Compositionality* states that the meaning of a compound expression is a function of the meaning of its parts and of the syntactic rules by which they are combined (e.g., Montague (1974), Dowty et al. (1981)). And Moilanen and Pulman (2007) develop a collection of composition rules to assign a sentiment value to individual expressions, clauses, or sentences. Their approach can be viewed as a type of structural inference, but their hand-written rules have not been empirically compared to learning-based alternatives, which one might expect to be more effective in handling some aspects of the polarity classification task.

In this paper, we begin to close the gap between learning-based approaches to expression-level polarity classification and those founded on compositional semantics: we present a novel learning-based approach that incorporates structural inference motivated by compositional semantics into the learning procedure.

Adopting the view point of compositional semantics, our working assumption is that the polarity of a sentiment-bearing expression can be determined in a two-step process: (1) assess the polarities of the constituents of the expression, and then (2) apply a relatively simple set of inference rules to combine them recursively. Rather than a rigid application of hand-written compositional inference rules, however, we hypothesize that an ideal solution to the expression-level polarity classification task will be a method that can exploit ideas from compositional semantics while providing the flexibility needed to handle the complexities of real-world natural language — exceptions, unknown words, missing semantic features, and inaccurate or missing rules. The learning-based approach proposed in this paper takes a first step in this direction.

In addition to the novel learning approach, this paper presents new insights for *content-word negators*, which we define as content words that can negate the polarity of neighboring words or constituents. (e.g., words such as "eliminated" in the example sentences). Unlike *function-word negators*, such as "not" or "never", content-word negators have been recognized and utilized less actively in previous work. (Notable exceptions include e.g., Niu et al. (2005), Wilson et al. (2005), and Moilanen and Pulman (2007).[2])

In our experiments, we compare learning- and non-learning-based approaches to expression-level polarity classification — with and without compositional semantics — and find that (1) simple heuristics based on compositional semantics outperform (89.7% in accuracy) other reasonable heuristics that do not incorporate compositional semantics (87.7%); they can also perform better than simple learning-based methods that do not incorporate compositional semantics (89.1%), (2) combining learning with the heuristic rules based on compositional semantics further improves the performance (90.7%), (3) content-word negators play an important role in determining the expression-level polarity, and, somewhat surprisingly, we find that (4) expression-level classification accuracy uniformly decreases as additional, potentially disambiguating, context is considered.

In what follows, we first explore heuristic-based approaches in §2, then we present learning-based approaches in §3. Next we present experimental results in §4, followed by related work in §5.

2 Heuristic-Based Methods

This section describes a set of heuristic-based methods for determining the polarity of a sentiment-bearing expression. Each assesses the polarity of the words or constituents using a polarity lexicon that indicates whether a word has positive or negative polarity, and finds negators in the given expression using a negator lexicon. The methods then infer the expression-level polarity using voting-based heuristics (§ 2.1) or heuristics that incorporate compositional semantics (§2.2). The lexicons are described

[2]See §5. Related Work for detailed discussion.

	VOTE	NEG(1)	NEG(N)	NEGEX(1)	NEGEX(N)	COMPO
type of negators	none	function-word		function-word & content-word		
maximum # of negations applied	0	1	n	1	n	n
scope of negators	N/A	over the entire expression				compositional

Table 1: Heuristic methods. (n refers to the number of negators found in a given expression.)

	Rules		Examples
1	Polarity(not_[arg1]) =	$\neg$ Polarity(arg1)	not [bad]$_{arg1}$.
2	Polarity([VP]_[NP]) =	Compose([VP], [NP])	[destroyed]$_{VP}$ [the terrorism]$_{NP}$.
3	Polarity([VP1]_to_[VP2]) =	Compose([VP1], [VP2])	[refused]$_{VP1}$ to [deceive]$_{VP2}$ the man.
4	Polarity([adj]_to_[VP]) =	Compose([adj], [VP])	[unlikely]$_{adj}$ to [destroy]$_{VP}$ the planet.
5	Polarity([NP1]_[IN]_[NP2]) =	Compose([NP1], [NP2])	[lack]$_{NP1}$ [of]$_{IN}$ [crime]$_{NP2}$ in rural areas.
6	Polarity([NP]_[VP]) =	Compose([VP], [NP])	[pollution]$_{NP}$ [has decreased]$_{VP}$.
7	Polarity([NP]_be_[adj]) =	Compose([adj], [NP])	[harm]$_{NP}$ is [minimal]$_{adj}$.

Definition of Compose(arg1, arg2)

For COMPOMC: (**COMPO**sition with **M**ajority **C**lass)	Compose(arg1, arg2) =
	 if (arg1 is a negator) then $\neg$ Polarity(arg2)
	 else if (Polarity(arg1) == Polarity(arg2)) then Polarity(arg1)
	 else the majority polarity of data

For COMPOPR: (**COMPO**sition with **PR**iority)	Compose(arg1, arg2) =
	 if (arg1 is a negator) then $\neg$ Polarity(arg2)
	 else Polarity(arg1)

Table 2: Compositional inference rules motivated by compositional semantics.

in §2.3.

2.1 Voting

We first explore five simple heuristics based on voting. VOTE is defined as the majority polarity vote by words in a given expression. That is, we count the number of positive polarity words and negative polarity words in a given expression, and assign the majority polarity to the expression. In the case of a tie, we default to the prevailing polarity of the data.

For NEG(1), we first determine the majority polarity vote as above, and then if the expression contains *any* function-word negator, flip the polarity of the majority vote once. NEG(N) is similar to NEG(1), except we flip the polarity of the majority vote n times after the majority vote, where n is the number of function-word negators in a given expression.

NEGEX(1) and NEGEX(N) are defined similarly as NEG(1) and NEG(N) above, except both function-word negators and content-word negators are considered as negators when flipping the polarity of the majority vote. See Table 1 for summary. Note that a word can be both a negator and have a negative prior polarity. For the purpose of voting, if a word is defined as a negator per the voting scheme, then that word does not participate in the majority vote.

For brevity, we refer to NEG(1) and NEG(N) collectively as NEG, and NEGEX(1) and NEGEX(N) collectively as NEGEX.

2.2 Compositional semantics

Whereas the heuristics above use voting-based inference, those below employ a set of hand-written rules motivated by compositional semantics. Table 2 shows the definition of the rules along with motivating examples. In order to apply a rule, we first detect a syntactic pattern (e.g., [destroyed]$_{VP}$ [the terrorism]$_{NP}$), then apply the *Compose* function as defined in Table 2 (e.g., Compose([destroyed], [the terrorism]) by rule #2).[3]

[3]Our implementation uses part-of-speech tags and function-words to coarsely determine the patterns. An implementation

Compose first checks whether the first argument is a negator, and if so, flips the polarity of the second argument. Otherwise, *Compose* resolves the polarities of its two arguments. Note that if the second argument is a negator, we do not flip the polarity of the first argument, because the first argument in general is not in the semantic scope of the negation.[4] Instead, we treat the second argument as a constituent with negative polarity.

We experiment with two variations of the *Compose* function depending on how conflicting polarities are resolved: COMPOMC uses a *Compose* function that defaults to the **M**ajority **C**lass of the polarity of the data,[5] while COMPOPR uses a *Compose* function that selects the polarity of the argument that has higher semantic **PR**iority. For brevity, we refer to COMPOPR and COMPOMC collectively as COMPO.

2.3 Lexicons

The polarity lexicon is initialized with the lexicon of Wilson et al. (2005) and then expanded using the General Inquirer dictionary.[6] In particular, a word contained in at least two of the following categories is considered as positive: POSITIV, PSTV, POSAFF, PLEASUR, VIRTUE, INCREAS, and a word contained in at least one of the following categories is considered as negative: NEGATIV, NGTV, NEGAFF, PAIN, VICE, HOSTILE, FAIL, ENLLOSS, WLBLOSS, TRANLOSS.

For the (function- and content-word) negator lexicon, we collect a handful of seed words as well as General Inquirer words that appear in either NOTLW or DECREAS category. Then we expand the list of content-negators using the synonym information of WordNet (Miller, 1995) to take a simple vote among senses.

based on parse trees might further improve the performance.

[4]Moilanen and Pulman (2007) provide more detailed discussion on the semantic scope of negations and the semantic priorities in resolving polarities.

[5]The majority polarity of the data we use for our experiments is negative.

[6]Available at http://www.wjh.harvard.edu/∼inquirer/. When consulting the General Inquirer dictionary, senses with less than 5% frequency and senses specific to an idiom are dropped.

3 Learning-Based Methods

While we expect that a set of hand-written heuristic rules motivated by compositional semantics can be effective for determining the polarity of a sentiment-bearing expression, we do not expect them to be perfect. Interpreting natural language is such a complex task that writing a perfect set of rules would be extremely challenging. Therefore, a more ideal solution would be a learning-based method that can exploit ideas from compositional semantics while providing the flexibility to the rigid application of the heuristic rules. To this end, we present a novel learning-based approach that incorporates inference rules inspired by compositional semantics into the learning procedure (§3.2). To assess the effect of compositional semantics in the learning-based methods, we also experiment with a simple classification approach that does not incorporate compositional semantics (§3.1). The details of these two approaches are elaborated in the following subsections.

3.1 Simple Classification (SC)

Given an expression x consisting of n words x_1, ..., x_n, the task is to determine the polarity $y \in \{positive, negative\}$ of x. In our simple binary classification approach, x is represented as a vector of features $\mathbf{f}(x)$, and the prediction y is given by $\text{argmax}_y \mathbf{w} \cdot \mathbf{f}(x, y)$, where $\mathbf{w}$ is a vector of parameters learned from training data. In our experiment, we use an online SVM algorithm called MIRA (Margin Infused Relaxed Algorithm) (Crammer and Singer, 2003)[7] for training.

For each x, we encode the following features:

- Lexical: We add every word x_i in x, and also add the lemma of x_i produced by the CASS partial parser toolkit (Abney, 1996).
- Dictionary: In order to mitigate the problem of unseen words in the test data, we add features that describe word categories based on the General Inquirer dictionary. We add this feature for each x_i that is not a stop word.
- Vote: We experiment with two variations of voting-related features: for SC-VOTE, we add

[7]We use the Java implementation of this algorithm available at http://www.seas.upenn.edu/∼strctlrn/StructLearn /StructLearn.html.

Simple Classification	Classification with Compositional Inference
$y \leftarrow \text{argmax}_y \text{ score}(y)$ $l \leftarrow \text{loss\_flat}(y^*, y)$ $\mathbf{w} \leftarrow \text{update}(\mathbf{w}, l, y^*, y)$	Find K best $\mathbf{z}$ and denote them as $\mathcal{Z} = \{\mathbf{z}^{(1)}, ..., \mathbf{z}^{(K)}\}$ $s.t. \forall\, i < j,\ \text{score}(\mathbf{z}^{(i)}) > \text{score}(\mathbf{z}^{(j)})$ $\mathbf{z}^{bad} \leftarrow \min_k \mathbf{z}^{(k)}\ s.t.\ \text{loss\_compo}(y^*, \mathbf{z}^{(k)}, x) > 0$ (if such $\mathbf{z}^{bad}$ not found in $\mathcal{Z}$, skip parameter update for this.) If $\text{loss\_compo}(y^*, \mathbf{z}^*, x) > 0$ $\mathbf{z}^{good} \leftarrow \min_k \mathbf{z}^{(k)}\ s.t.\ \text{loss\_compo}(y^*, \mathbf{z}^{(k)}, x) = 0$ $z^* \leftarrow \mathbf{z}^{good}$ (if such $\mathbf{z}^{good}$ not found in $\mathcal{Z}$, stick to the original z^*.) $l \leftarrow \text{loss\_compo}(y^*, \mathbf{z}^{bad}, x) - \text{loss\_compo}(y^*, \mathbf{z}^*, x)$ $\mathbf{w} \leftarrow \text{update}(\mathbf{w}, l, \mathbf{z}^*, \mathbf{z}^{bad})$
Definitions of score functions and loss functions	
$\text{score}(y) := \mathbf{w} \cdot \mathbf{f}(x, y)$ $\text{loss\_flat}(y^*, y) := \text{if } (y^* = y)\ 0 \text{ else } 1$	$\text{score}(\mathbf{z}) := \sum_i \text{score}(z_i) := \sum_i \mathbf{w} \cdot \mathbf{f}(x, z_i, i)$ $\text{loss\_compo}(y^*, \mathbf{z}, x) := \text{if } (y^* = \mathcal{C}(x, \mathbf{z}))\ 0 \text{ else } 1$

Figure 1: Training procedures. $y^* \in \{positive, negative\}$ denotes the true label for a given expression $x = x_1, ..., x_n$. $\mathbf{z}^*$ denotes the pseudo gold standard for hidden variables $\mathbf{z}$.

a feature that indicates the dominant polarity of words in the given expression, without considering the effect of negators. For SC-NEGEX, we count the number of content-word negators as well as function-word negators to determine whether the final polarity should be flipped. Then we add a conjunctive feature that indicates the dominant polarity together with whether the final polarity should be flipped. For brevity, we refer to SC-VOTE and SC-NEGEX collectively as SC.

Notice that in this simple binary classification setting, it is inherently difficult to capture the compositional structure among words in x, because $\mathbf{f}(x, y)$ is merely a flat bag of features, and the prediction is governed simply by the dot product of $\mathbf{f}(x, y)$ and the parameter vector w.

3.2 Classification with Compositional Inference (CCI)

Next, instead of determining y directly from x, we introduce hidden variables $\mathbf{z} = (z_1, ..., z_n)$ as intermediate decision variables, where $z_i \in \{positive, negative, negator, none\}$, so that z_i represents whether x_i is a word with positive/negative polarity, or a negator, or none of the above. For simplicity, we let each intermediate decision variable z_i (a) be determined independently from other intermediate decision variables, and (b) depend only on the input x, so that $z_i = \text{argmax}_{z_i} \mathbf{w} \cdot \mathbf{f}(x, z_i, i)$, where $\mathbf{f}(x, z_i, i)$ is the feature vector encoding around the ith word (described on the next page). Once we determine the intermediate decision variables, we apply the heuristic rules motivated by compositional semantics (from Table 2) in order to obtain the final polarity y of x. That is, $y = \mathcal{C}(x, \mathbf{z})$, where $\mathcal{C}$ is the function that applies the compositional inference, either COMPOPR or COMPOMC.

For training, there are two issues we need to handle: the first issue is dealing with the hidden variables z. Because the structure of compositional inference $\mathcal{C}$ does not allow dynamic programming, it is intractable to perform exact expectation-maximization style training that requires enumerating all possible values of the hidden variables $\mathbf{z}$. Instead, we propose a simple and tractable training

```
For each token $x_i$,
  if $x_i$ is a word in the negator lexicon
    then $z_i^* \leftarrow negator$
  else if $x_i$ is in the polarity lexicon as negative
    then $z_i^* \leftarrow negative$
  else if $x_i$ is in the polarity lexicon as positive
    then $z_i^* \leftarrow positive$
  else
    then $z_i^* \leftarrow none$
```

Figure 2: Constructing Soft Gold Standard $\mathbf{z}^*$

rule based on the creation of a *soft* gold standard for $\mathbf{z}$. In particular, we exploit the fact that in our task, we can automatically construct a reasonably accurate gold standard for $\mathbf{z}$, denoted as $\mathbf{z}^*$: as shown in Figure 2, we simply rely on the negator and polarity lexicons. Because $\mathbf{z}^*$ is not always correct, we allow the training procedure to replace $\mathbf{z}^*$ with potentially better assignments as learning proceeds: in the event that the soft gold standard $\mathbf{z}^*$ leads to an incorrect prediction, we search for an assignment that leads to a correct prediction to replace $\mathbf{z}^*$. The exact procedure is given in Figure 1, and will be discussed again shortly.

Figure 1 shows how we modify the parameter update rule of MIRA (Crammer and Singer, 2003) to reflect the aspect of compositional inference. In the event that the soft gold standard $\mathbf{z}^*$ leads to an incorrect prediction, we search for $\mathbf{z}^{good}$, the assignment with highest score that leads to a correct prediction, and replace $\mathbf{z}^*$ with $\mathbf{z}^{good}$. In the event of no such $\mathbf{z}^{good}$ being found among the K-best assignments of $\mathbf{z}$, we stick with $\mathbf{z}^*$.

The second issue is finding the assignment of $\mathbf{z}$ with the highest score$(\mathbf{z}) = \sum_i \mathbf{w} \cdot \mathbf{f}(x, z_i, i)$ that leads to an incorrect prediction $y = \mathcal{C}(x, \mathbf{z})$. Because the structure of compositional inference $\mathcal{C}$ does not allow dynamic programming, finding such an assignment is again intractable. We resort to enumerating only over K-best assignments instead. If none of the K-best assignments of $\mathbf{z}$ leads to an incorrect prediction y, then we skip the training instance for parameter update.

Features. For each x_i in x, we encode the following features:

- Lexical: We include the current word x_i as well as the lemma of x_i produced by CASS partial parser toolkit (Abney, 1996). We also add a boolean feature to indicate whether the current word is a stop word.
- Dictionary: In order to mitigate the problem with unseen words in the test data, we add features that describe word categories based on the General Inquirer dictionary. We add this feature for each x_i that is not a stop word. We also add a number of boolean features that provide following properties of x_i using the polarity lexicon and the negator lexicon:
  - whether x_i is a function-word negator
  - whether x_i is a content-word negator
  - whether x_i is a negator of any kind
  - the polarity of x_i according to Wilson et al. (2005)'s polarity lexicon
  - the polarity of x_i according to the lexicon derived from the General Inquirer dictionary
  - conjunction of the above two features
- Vote: We encode the same vote feature that we use for SC-NEGEX described in § 3.1.

As in the heuristic-based compositional semantics approach (§ 2.2), we experiment with two variations of this learning-based approach: CCI-COMPOPR and CCI-COMPOMC, whose compositional inference rules are COMPOPR and COMPOMC respectively. For brevity, we refer to both variations collectively as CCI-COMPO.

4 Experiments

The experiments below evaluate our heuristic- and learning-based methods for subsentential sentiment analysis (§ 4.1). In addition, we explore the role of context by expanding the boundaries of the sentiment-bearing expressions (§ 4.2).

4.1 Evaluation with given boundaries

For evaluation, we use the Multi-Perspective Question Answering (MPQA) corpus (Wiebe et al., 2005), which consists of 535 newswire documents manually annotated with phrase-level subjectivity information. We evaluate on all strong (i.e., intensity of expression is 'medium' or higher), sentiment-bearing (i.e., polarity is 'positive' or 'negative') expressions.[8] As a result, we can assume the boundaries of the expressions are given. Performance is reported using 10-fold cross-validation on 400 documents; a separate 135 documents were used as a development set. Based on pilot experiments on the development data, we set parameters for MIRA as follows: slack variable to 0.5, and the number of incorrect labels (constraints) for each parameter update to 1. The number of iterations (epochs) for training is set to 1 for simple classification, and to 4

[8]We discard expressions with confidence marked as 'uncertain'.

Heuristic-Based							Learning-Based			
VOTE	NEG (1)	NEG (N)	NEG EX (1)	NEG EX (N)	COMPO MC	COMPO PR	SC VOTE	SC NEG EX	CCI COMPO MC	CCI COMPO PR
86.5	82.0	82.2	87.7	87.7	89.7	89.4	88.5	89.1	90.6	90.7

Table 3: Performance (in accuracy) on MPQA dataset.

Data	Heuristic-Based							Learning-Based			
	VOTE	NEG (1)	NEG (N)	NEG EX (1)	NEG EX (N)	COMPO MC	COMPO PR	SC VOTE	SC NEG EX	CCI COMPO MC	CCI COMPO PR
[-0,+0]	86.5	82.0	82.2	87.7	87.7	89.7	89.4	88.5	89.1	90.6	90.7
[-1,+1]	86.4	81.0	81.2	87.2	87.2	89.3	89.0	88.3	88.4	89.5	89.4
[-5,+5]	85.9	79.0	79.4	85.7	85.6	88.2	88.0	86.4	87.1	88.7	88.7
[-∞,+∞]	85.3	75.8	76.9	83.9	83.9	87.0	86.9	85.8	85.8	87.3	87.5

Table 4: Performance (in accuracy) on MPQA data set with varying boundaries of expressions.

for classification with compositional inference. We use $K = 20$ for classification with compositional inference.

Results. Performance is reported in Table 3. Interestingly, the heuristic-based methods NEG ($\sim 82.2\%$) that only consider function-word negators perform even worse than VOTE (86.5%), which does not consider negators. On the other hand, the NEGEX methods (87.7%) that do consider content-word negators as well as function-word negators perform better than VOTE. This confirms the importance of content-word negators for determining the polarities of expressions. The heuristic-based methods motivated by compositional semantics COMPO further improve the performance over NEGEX, achieving up to 89.7% accuracy. In fact, these heuristics perform even better than the SC learning-based methods ($\sim 89.1\%$). This shows that heuristics that take into account the compositional structure of the expression can perform better than learning-based methods that do not exploit such structure.

Finally, the learning-based methods that incorporate compositional inference CCI-COMPO ($\sim 90.7\%$) perform better than all of the previous methods. The difference between CCI-COMPOPR (90.7%) and SC-NEGEX (89.1%) is statistically significant at the .05 level by paired t-test. The difference between COMPO and any other heuristic that is not based on computational semantics is also statistically significant. In addition, the difference between CCICOMPOPR (learning-based) and COMPOMC (non-learning-based) is statistically significant, as is the difference between NEGEX and VOTE.

4.2 Evaluation with noisy boundaries

One might wonder whether employing additional context outside the annotated expression boundaries could further improve the performance. Indeed, conventional wisdom would say that it is necessary to employ such contextual information (e.g., Wilson et al. (2005)). In any case, it is important to determine whether our results will apply to more real-world settings where human-annotated expression boundaries are not available.

To address these questions, we gradually relax our previous assumption that the exact boundaries of expressions are given: for each annotation boundary, we expand the boundary by x words for each direction, up to sentence boundaries, where $x \in \{1, 5, \infty\}$. We stop expanding the boundary if it will collide with the boundary of an expression with a different polarity, so that we can consistently recover the expression-level gold standard for evaluation. This expansion is applied to both the training and test data, and the performance is reported in Table 4. From this experiment, we make the following observations:

- Expanding the boundaries hurts the performance for any method. This shows that most of relevant context for judging the polarity is contained within the expression boundaries, and motivates the task of finding the boundaries of opinion expressions.

- The NegEx methods perform better than Vote only when the expression boundaries are reasonably accurate. When the expression boundaries are expanded up to sentence boundaries, they perform worse than Vote. We conjecture this is because the scope of negators tends to be limited to inside of expression boundaries.
- The Compo methods always perform better than any other heuristic-based methods. And their performance does not decrease as steeply as the NegEx methods as the expression boundaries expand. We conjecture this is because methods based on compositional semantics can handle the scope of negators more adequately.
- Among the learning-based methods, those that involve compositional inference (CCI-Compo) always perform better than those that do not (SC) for any boundaries. And learning with compositional inference tend to perform better than the rigid application of heuristic rules (Compo), although the relative performance gain decreases once the boundaries are relaxed.

5 Related Work

The task focused on in this paper is similar to that of Wilson et al. (2005) in that the general goal of the task is to determine the polarity in context at a sub-sentence level. However, Wilson et al. (2005) formulated the task differently by limiting their evaluation to individual words that appear in their polarity lexicon. Also, their approach was based on a flat bag of features, and only a few examples of what we call content-word negators were employed.

Our use of compositional semantics for the task of polarity classification is preceded by Moilanen and Pulman (2007), but our work differs in that we integrate the key idea of compositional semantics into learning-based methods, and that we perform empirical comparisons among reasonable alternative approaches. For comparison, we evaluated our approaches on the polarity classification task from SemEval-07 (Strapparava and Mihalcea, 2007). We achieve 88.6% accuracy with CompoPR, 90.1% with SCNegEx, and 87.6% with CCICompoMC.[9] There are a number of possible reasons for our lower performance vs. Moilanen and Pulman (2007) on this data set. First, SemEval-07 does not include a training data set for this task, so we use 400 documents from the MPQA corpus instead. In addition, the SemEval-07 data is very different from the MPQA data in that (1) the polarity annotation is given only at the sentence level, (2) the sentences are shorter, with simpler structure, and not as many negators as the MPQA sentences, and (3) there are many more instances with positive polarity than in the MPQA corpus.

Nairn et al. (2006) also employ a "polarity" propagation algorithm in their approach to the semantic interpretation of implicatives. However, their notion of polarity is quite different from that assumed here and in the literature on sentiment analysis. In particular, it refers to the degree of "commitment" of the author to the truth or falsity of a complement clause for a textual entailment task.

McDonald et al. (2007) use a structured model to determine the sentence-level polarity and the document-level polarity simultaneously. But decisions at each sentence level does not consider structural inference within the sentence.

Among the studies that examined content-word negators, Niu et al. (2005) manually collected a small set of such words (referred as "words that change phases"), but their lexicon was designed mainly for the medical domain and the type of negators was rather limited. Wilson et al. (2005) also manually collected a handful of content-word negators (referred as "general polarity shifters"), but not extensively. Moilanen and Pulman (2007) collected a more extensive set of negators semi-automatically using WordNet 2.1, but the empirical effect of such words was not explicitly investigated.

[9]For lack of space, we only report our performance on instances with strong intensities as defined in Moilanen and Pulman (2007), which amounts to only 208 test instances. The cross-validation set of MPQA contains $4.9k$ instances.

6 Conclusion

In this paper, we consider the task of determining the polarity of a sentiment-bearing expression, considering the effect of interactions among words or constituents in light of compositional semantics. We presented a novel learning-based approach that incorporates structural inference motivated by compositional semantics into the learning procedure. Our approach can be considered as a small step toward bridging the gap between computational semantics and machine learning methods. Our experimental results suggest that this direction of research is promising. Future research includes an approach that learns the compositional inference rules from data.

Acknowledgments

This work was supported in part by National Science Foundation Grants BCS-0624277 and IIS-0535099 and by Department of Homeland Security Grant N0014-07-1-0152. We also thank Eric Breck, Lillian Lee, Mats Rooth, the members of the Cornell NLP reading seminar, and the EMNLP reviewers for insightful comments on the submitted version of the paper.

References

Steven Abney. 1996. Partial parsing via finite-state cascades. *Journal of Natural Language Engineering*, 2(4):337344.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *JMLR* 3:951.

David R. Dowty, Robert E. Wall and Stanley Peters. 1981. *Introduction to Montague Semantics.*

Andrea Esuli and Fabrizio Sebastiani. 2006. SentiWordNet: A Publicly Available Lexical Resource for Opinion Mining. In *Proceedings of 5th Conference on Language Resources and Evaluation (LREC),.*

Percy Liang, Hal Daumé III and Dan Klein. 2008. Structure Compilation: Trading Structure for Features. In *International Conference on Machine Learning*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-2004).*

Alistair Kennedy and Diana Inkpen. 2005. Sentiment Classification of Movie and Product Reviews Using Contextual Valence Shifters. In *Proceedings of FINEXIN 2005, Workshop on the Analysis of Informal and Formal Information Exchange during Negotiations.*

Soo-Min Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of COLING.*

Ryan McDonald, Kerry Hannan, Tyler Neylon, Mike Wells and Jeff Reynar. 2007. Structured Models for Fine-to-Coarse Sentiment Analysis. In *Proceedings of Association for Computational Linguistics (ACL)* .

George A. Miller. 1995. WordNet: a lexical database for English. In *Communications of the ACM*, 38(11):3941

Richard Montague. 1974. Formal Philosophy; Selected papers of Richard Montague. Yale University Press.

Karo Moilanen and Stephen Pulman. 2007. Sentiment Composition. In *Proceedings of Recent Advances in Natural Language Processing (RANLP 2007).*

Rowan Nairn, Cleo Condoravdi and Lauri Karttunen 2006. Computing relative polarity for textual inference. In *Inference in Computational Semantics (ICoS-5).*

Yun Niu, Xiaodan Zhu, Jianhua Li and Graeme Hirst. 2005. Analysis of polarity information in medical text. In *Proceedings of the American Medical Informatics Association 2005 Annual Symposium (AMIA).*

Livia Polanyi and Annie Zaenen. 2004. Contextual lexical valence shifters. In *Exploring Attitude and Affect in Text: Theories and Applications: Papers from the 2004 Spring Symposium, AAAI.*

Mostafa Shaikh, Helmut Prendinger and Mitsuru Ishizuka. 2007. Assessing sentiment of text by semantic dependency and contextual valence analysis. In *Proc 2nd Int'l Conf on Affective Computing and Intelligent Interaction (ACII'07).*

Carlo Strapparava and Rada Mihalcea. 2007. Semeval-2007 task 14: Affective text. In *Proceedings of SemEval.*

Janyce Wiebe, Theresa Wilson and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. In *Language Resources and Evaluation (formerly Computers and the Humanities), 39(2-3):165210.*

Theresa Wilson, Janyce Wiebe and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of HLT/EMNLP.*

A Phrase-Based Alignment Model for Natural Language Inference

Bill MacCartney, Michel Galley, Christopher D. Manning
Natural Language Processing Group, Stanford University
{wcmac,mgalley,manning}@stanford.edu

Abstract

The alignment problem—establishing links between corresponding phrases in two related sentences—is as important in natural language inference (NLI) as it is in machine translation (MT). But the tools and techniques of MT alignment do not readily transfer to NLI, where one cannot assume semantic equivalence, and for which large volumes of bitext are lacking. We present a new NLI aligner, the MANLI system, designed to address these challenges. It uses a phrase-based alignment representation, exploits external lexical resources, and capitalizes on a new set of supervised training data. We compare the performance of MANLI to existing NLI and MT aligners on an NLI alignment task over the well-known Recognizing Textual Entailment data. We show that MANLI significantly outperforms existing aligners, achieving gains of 6.2% in F_1 over a representative NLI aligner and 10.5% over GIZA++.

1 Introduction

The problem of natural language inference (NLI) is to determine whether a natural-language hypothesis H can reasonably be inferred from a given premise text P. In order to recognize that *Kennedy was killed* can be inferred from *JFK was assassinated*, one must first recognize the correspondence between *Kennedy* and *JFK*, and between *killed* and *assassinated*. Consequently, most current approaches to NLI rely, implicitly or explicitly, on a facility for *alignment*—that is, establishing links between corresponding entities and predicates in P and H. Recent entries in the annual Recognizing Textual Entailment (RTE) competition (Dagan et al., 2005) have addressed the alignment problem in a variety of ways, though often without distinguishing it as a separate subproblem. Glickman et al. (2005) and Jijkoun and de Rijke (2005), among others, have explored approaches based on measuring the degree of lexical overlap between bags of words. While ignoring structure, such methods depend on matching each word in H to the word in P with which it is most similar—in effect, an alignment. At the other extreme, Tatu and Moldovan (2007) and Bar-Haim et al. (2007) have formulated the inference problem as analogous to proof search, using inferential rules which encode (among other things) knowledge of lexical relatedness. In such approaches, the correspondence between the words of P and H is implicit in the steps of the proof.

Increasingly, however, the most successful RTE systems have made the alignment problem explicit. Marsi and Krahmer (2005) and MacCartney et al. (2006) first advocated pipelined system architectures containing a distinct alignment component, a strategy crucial to the top-performing systems of Hickl et al. (2006) and Hickl and Bensley (2007). However, each of these systems has pursued alignment in idiosyncratic and poorly-documented ways, often using proprietary data, making comparisons and further development difficult.

In this paper we undertake the first systematic study of alignment for NLI. We propose a new NLI alignment system which uses a phrase-based representation of alignment, exploits external resources for knowledge of semantic relatedness, and capitalizes on the recent appearance of new supervised training data for NLI alignment. In addition, we examine the relation between NLI alignment and MT alignment, and investigate whether existing MT aligners can usefully be applied in the NLI setting.

2 NLI alignment vs. MT alignment

The alignment problem is familiar in machine translation (MT), where recognizing that *she came* is a good translation for *elle est venue* requires establishing a correspondence between *she* and *elle*, and between *came* and *est venue*. The MT community has developed not only an extensive literature on alignment (Brown et al., 1993; Vogel et al., 1996; Marcu and Wong, 2002; DeNero et al., 2006), but also standard, proven alignment tools such as GIZA++ (Och and Ney, 2003). Can off-the-shelf MT aligners be applied to NLI? There is reason to be doubtful. Alignment for NLI differs from alignment for MT in several important respects, including:

1. Most obviously, it is monolingual rather than cross-lingual, opening the door to utilizing abundant (monolingual) sources of information on semantic relatedness, such as WordNet.
2. It is intrinsically asymmetric: P is often much longer than H, and commonly contains phrases or clauses which have no counterpart in H.
3. Indeed, one cannot assume even approximate semantic equivalence—usually a given in MT. Because NLI problems include both valid and invalid inferences, the semantic content of H may diverge substantially from P. An NLI aligner must be designed to accommodate frequent unaligned words and phrases.
4. Little training data is available. MT alignment models are typically trained in unsupervised fashion, inducing lexical correspondences from massive quantities of sentence-aligned bitexts. While NLI aligners could in principle do the same, large volumes of suitable data are lacking. NLI aligners must therefore depend on smaller quantities of supervised training data, supplemented by external lexical resources. Conversely, while existing MT aligners can make use of dictionaries, they are not designed to harness other sources of information on degrees of semantic relatedness.

Consequently, the tools and techniques of MT alignment may not transfer readily to NLI alignment. We investigate the matter empirically in section 5.2.

3 Data

Until recently, research on alignment for NLI has been hampered by a paucity of high-quality, publicly available data from which to learn. Happily, that has begun to change, with the release by Microsoft Research (MSR) of human-generated alignment annotations (Brockett, 2007) for inference problems from the second Recognizing Textual Entailment (RTE2) challenge (Bar-Haim et al., 2006). To our knowledge, this work is the first to exploit this data for training and evaluation of NLI alignment models.

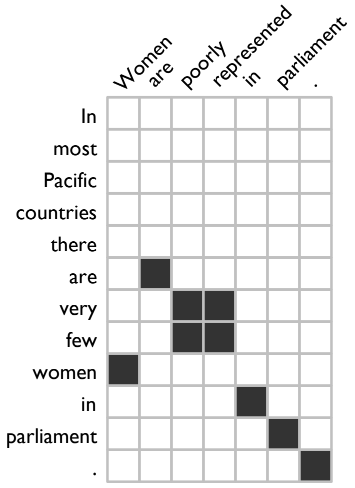


Figure 1: The MSR gold-standard alignment for problem 116 from the RTE2 development set.

The RTE2 data consists of a development set and a test set, each containing 800 inference problems. Each problem consists of a premise and a hypothesis. The premises contain 29 words on average; the hypotheses, 11 words. Each problem is marked as a valid or invalid inference (50% each); however, these annotations are ignored during alignment, since they would not be available during testing of a complete NLI system.

The MSR annotations use an alignment representation which is token-based, but many-to-many, and thus allows implicit alignment of multi-word phrases. Figure 1 shows an example in which *very few* has been aligned with *poorly represented*.

In the MSR data, every alignment link is marked as SURE or POSSIBLE. In making this distinction, the annotators have followed a convention common in MT, which permits alignment precision to be measured against both SURE and POSSIBLE links, while recall is measured against only SURE links. In this work, however, we have chosen to ignore POSSIBLE links, embracing the argument made by (Fraser and Marcu, 2007) that their use has impeded progress in MT alignment models, and that SURE-only annotation is to be preferred.

Each RTE2 problem was independently annotated by three people, following carefully designed annotation guidelines. Inter-annotator agreement was high: Brockett (2007) reports Fleiss' kappa[1] scores of about 0.73 ("substantial agreement") for mappings from H tokens to P tokens; and all three annotators agreed on $\sim$70% of proposed links, while at least two of three agreed on more than 99.7% of proposed links,[2] attesting to the high quality of the annotation data. For this work, we merged the three independent annotations, using majority rule,[3] to obtain a gold-standard annotation containing an average of 7.3 links per RTE problem.

4 The MANLI aligner

In this section, we describe the MANLI aligner, a new alignment system designed expressly for NLI alignment. The MANLI system consists of four elements: (1) a phrase-based representation of alignment, (2) a feature-based linear scoring function for alignments, (3) a decoder which uses simulated annealing to find high-scoring alignments, and (4) perceptron learning to optimize feature weights.

4.1 A phrase-based alignment representation

MANLI uses an alignment representation which is intrinsically phrase-based. (Following the usage common in MT, we use "phrase" to mean any contiguous span of tokens, not necessarily corresponding to a syntactic phrase.) We represent an alignment E between a premise P and a hypothesis H as a set of phrase edits $\{e_1, e_2, \ldots\}$, each belonging to one of four types:

- an EQ edit connects a phrase in P with an equal (by word lemmas) phrase in H
- a SUB edit connects a phrase in P with an unequal phrase in H
- a DEL edit covers an unaligned phrase in P
- an INS edit covers an unaligned phrase in H

For example, the alignment shown in figure 1 can be represented by the set $\{$DEL(*In*$_1$), DEL(*most*$_2$), DEL(*Pacific*$_3$), DEL(*countries*$_4$), DEL(*there*$_5$), EQ(*are*$_6$, *are*$_2$), SUB(*very*$_7$ *few*$_8$, *poorly*$_3$ *represented*$_4$), EQ(*women*$_9$, *Women*$_1$), EQ(*in*$_{10}$, *in*$_5$), EQ(*parliament*$_{11}$, *parliament*$_6$), EQ(.$_{12}$, .$_7$)$\}$.[4]

Alignments are constrained to be one-to-one at the phrase level: every token in P and H belongs to exactly one phrase, which participates in exactly one edit (possibly DEL or INS). However, the phrase representation permits alignments which are many-to-many at the token level. In fact, this is the chief motivation for the phrase-based representation: we can align *very few* and *poorly represented* as units, without being forced to make an arbitrary choice as to which word goes with which word. Moreover, our scoring function can make use of lexical resources which have information about semantic relatedness of multi-word phrases, not merely individual words.

About 23% of the MSR gold-standard alignments are not one-to-one (at the token level), and are therefore technically unreachable for MANLI, which is constrained to generate one-to-one alignments. However, by merging contiguous token links into phrase edits of size > 1, most MSR alignments (about 92%) can be straightforwardly converted into MANLI-reachable alignments. For the purpose of model training (but *not* for the evaluation described in section 5.4), we generated a version of the MSR data in which all alignments were converted to MANLI-reachable form.[5]

4.2 A feature-based scoring function

To score alignments, we use a simple feature-based linear scoring function, in which the score of an alignment is the sum of the scores of the edits it contains (including not only SUB and EQ edits, but also DEL and INS edits), and the score of an edit is the dot product of a vector encoding its features and a vector of weights. If E is a set of edits constituting

[1]Fleiss' kappa generalizes Cohen's kappa to the case where there are more than two annotators.

[2]The SURE/POSSIBLE distinction is taken as significant in computing all these figures.

[3]The handful of three-way disagreements were treated as POSSIBLE links, and thus were not used here.

[4]DEL and INS edits of size > 1 are possible in principle, but are not used in our training data.

[5]About 8% of the MSR alignments contain non-contiguous links, most commonly because P contains two references to an entity (e.g., *Christian Democrats* and *CDU*) which are both linked to a reference to the same entity in H (e.g., *Christian Democratic Union*). In such cases, one or more links must be eliminated to achieve a MANLI-reachable alignment. We used a string-similarity heuristic to break such conflicts, but were obliged to make an arbitrary choice in about 2% of cases.

an alignment, and $\mathbf{\Phi}$ is a vector of feature functions, the score s is given by:

$$s(E) = \sum_{e \in E} s(e) = \sum_{e \in E} \mathbf{w} \cdot \mathbf{\Phi}(e)$$

We'll explain how the feature weights $\mathbf{w}$ are set in section 4.4. The features used to characterize each edit are as follows:

Edit type features. We begin with boolean features encoding the type of each edit. We expect EQs to score higher than SUBs, and (since P is commonly longer than H) DELs to score higher than INSs.

Phrase features. Next, we have features which encode the sizes of the phrases involved in the edit, and whether these phrases are non-constituents (in syntactic parses of the sentences involved).

Lexical similarity feature. For SUB edits, a very important feature represents the lexical similarity of the substituends, as a real value in $[0, 1]$. This similarity score is computed as a max over a number of component scoring functions, some based on external lexical resources, including:

- various string similarity functions, of which most are applied to word lemmas
- measures of synonymy, hypernymy, antonymy, and semantic relatedness, including a widely-used measure due to Jiang and Conrath (1997), based on manually constructed lexical resources such as WordNet and NomBank
- a function based on the well-known distributional similarity metric of Lin (1998), which automatically infers similarity of words and phrases from their distributions in a very large corpus of English text

The ability to leverage external lexical resources—both manually and automatically constructed—is critical to the success of MANLI.

Contextual features. Even when the lexical similarity for a SUB edit is high, it may not be a good match. If P or H contains multiple occurrences of the same word—which happens frequently with function words, and occasionally with content words—lexical similarity may not suffice to determine the right match. To remedy this, we introduce contextual features for SUB and EQ edits. A real-valued *distortion* feature measures the difference between the relative positions of the substituends within their respective sentences, while boolean *matching neighbors* features indicate whether the tokens before and after the substituends are equal or similar.

```
Inputs
  • an alignment problem $\langle P, H \rangle$
  • a number of iterations $N$ (e.g. 100)
  • initial temperature $T_0$ (e.g. 40) and multiplier $r$ (e.g. 0.9)
  • a bound on edit size $max$ (e.g. 6)
  • an alignment scoring function, SCORE($E$)
Initialize
  • Let $E$ be an "empty" alignment for $\langle P, H \rangle$ (containing
    only DEL and INS edits, no EQ or SUB edits)
  • Set $\hat{E} = E$
Repeat for $i$ = 1 to $N$
  • Let $\{F_1, F_2, ...\}$ be the set of possible successors of $E$.
    To generate this set:
      – Consider every possible edit $f$ up to size $max$
      – Let $C(E, f)$ be the set of edits in $E$ which "conflict" with $f$
        (i.e., involve at least some of the same tokens as $f$)
      – Let $F = E \cup \{f\} \setminus C(E, f)$
  • Let $s(F)$ be a map from successors of $E$ to scores generated by SCORE
  • Set $p(F) = \exp s(F)$, and then normalize $p(F)$, transforming
    the score map to a probability distribution
  • Set $T_i = r \cdot T_{i-1}$
  • Set $p(F) = p(F)^{1/T_i}$, smoothing or sharpening $p(F)$
  • Renormalize $p(F)$
  • Choose a new value for $E$ by sampling from $p(F)$
  • If SCORE($E$) > SCORE($\hat{E}$), set $\hat{E}$ = $E$
Return $\hat{E}$
```

Figure 2: The MANLI-ALIGN algorithm

4.3 Decoding using simulated annealing

The problem of decoding—that is, finding a high-scoring alignment for a particular inference problem—is made more complex by our choice of a phrase-based alignment representation. For a model which uses a token-based representation (say, one which simply maps H tokens to P tokens), decoding is trivial, since each token can be aligned independently of its neighbors. (This is the case for the bag-of-words aligner described in section 5.1.) But with a phrase-based representation, things are more complicated. The segmentation into phrases is not given in advance, and every phrase pair considered for alignment must be consistent with its neighbors with respect to segmentation. Consequently, the decoding problem cannot be factored into a number of independent decisions.

To address this difficulty, we have devised a stochastic alignment algorithm, MANLI-ALIGN (figure 2), which uses a simulated annealing strategy. Beginning from an arbitrary alignment, we make a series of local steps, at each iteration sampling from a set of possible successors according to scores assigned by our scoring function. The sampling is controlled by a "temperature" which falls over time. At the beginning of the process, successors are sampled with nearly uniform probability, which helps to ensure that the space of possibilities is explored and local maxima are avoided. As the temperature falls, there is a ever-stronger bias toward high-scoring successors, so that the algorithm converges on a near-optimal alignment. Clever use of memoization helps to ensure that computational costs remain manageable. Using the parameter values suggested in figure 2, aligning an average RTE problem takes about two seconds.

While MANLI-ALIGN is not guaranteed to produce optimal alignments, there is reason to believe that it usually comes very close. After training, the alignment found by MANLI scored at least as high as the gold alignment for 99.6% of RTE problems.[6]

4.4 Perceptron learning

To tune the parameters **w** of the model, we use an adaptation of the averaged perceptron algorithm (Collins, 2002), which has proven successful on a range of NLP tasks. The algorithm is shown in figure 3. After initializing **w** to 0, we perform N training epochs. (Our experiments used $N = 50$.) In each epoch, we iterate through the training data, updating the weight vector at each training example according to the difference between the features of the target alignment and the features of the alignment produced by the decoder using the current weight vector. The size of the update is controlled by a learning rate which decreases over time. At the end of each epoch, the weight vector is normalized and stored. The final result is the average of the stored weight vectors, omitting vectors from a fixed number of epochs at the beginning of the run (which tend to be of poor quality). Using the parameter values suggested in figure 3, training runs on the RTE2 development set required about 20 hours.

```
Inputs
  • training problems $\langle P_j, H_j \rangle$, $j = 1..n$
  • corresponding gold-standard alignments $E_j$
  • a number of learning epochs $N$ (e.g. 50)
  • a "burn-in" period $N_0 < N$ (e.g. 10)
  • initial learning rate $R_0$ (e.g. 1) and multiplier $r$ (e.g. 0.8)
  • a vector of feature functions $\mathbf{\Phi}(E)$
  • an alignment algorithm ALIGN($P, H; \mathbf{w}$) which finds a
    good alignment for $\langle P, H \rangle$ using weight vector w
Initialize
  • Set $\mathbf{w} = 0$
Repeat for $i$ = 1 to $N$
  • Set $R_i = r \cdot R_{i-1}$, reducing the learning rate
  • Randomly shuffle the training problems
  • For $j$ = 1 to $n$:
      – Set $\hat{E}_j$ = ALIGN($P_j, H_j; \mathbf{w}$)
      – Set $\mathbf{w} = \mathbf{w} + R_i \cdot (\mathbf{\Phi}(E_j) - \mathbf{\Phi}(\hat{E}_j))$
  • Set $\mathbf{w} = \mathbf{w}/\|\mathbf{w}\|_2$ (L2 normalization)
  • Set $\mathbf{w}[i] = \mathbf{w}$, storing the weight vector for this epoch
Return an averaged weight vector:
  • $\mathbf{w}_{avg} = 1/(N - N_0) \sum_{i=N_0+1}^{N} \mathbf{w}[i]$
```

Figure 3: The MANLI-LEARN algorithm

5 Evaluating aligners on MSR data

In this section, we describe experiments designed to evaluate the performance of various alignment systems on the MSR gold-standard data described in section 3. For each system, we report precision, recall, and F-measure (F_1).[7] Note that these are macro-averaged statistics, computed per problem by counting aligned token pairs,[8] and then averaged over all problems in a problem set.[9] We also re-

[6]This figure is based on the MANLI-reachable version of the gold-standard data described in section 4.1. For the raw gold-standard data, the figure is 88.1%. The difference is almost entirely attributable to unreachable gold alignments, which tend to score higher simply because they contain more edits (and because the learned weights are mostly positive).

[7]MT researchers conventionally report results in terms of alignment error rate (AER). Since we use only SURE links in the gold-standard data (see section 3), AER is equivalent to $1 - F_1$.

[8]For phrase-based alignments like those generated by MANLI, two tokens are considered to be aligned iff they are contained within phrases which are aligned.

[9]MT evaluations conventionally use micro-averaging, which gives greater weight to problems containing more aligned pairs. This makes sense in MT, where the purpose of alignment is to induce phrase tables. But in NLI, where the ultimate goal is to maximize the number of inference problems answered correctly, it is more fitting to give all problems equal weight, and so we macro-average. We have also generated all results using micro-averaging, and found that the relative comparisons are

port the exact match rate, that is, the proportion of problems in which the guessed alignment exactly matches the gold alignment. The results are summarized in table 1.

5.1 A robust baseline: the bag-of-words aligner

As a baseline, we use a simple alignment algorithm inspired by the lexical entailment model of Glickman et al. (2005), and similar to the simple heuristic model described in (Och and Ney, 2003). Each hypothesis word h is aligned to the premise word p to which it is most similar, according to a lexical similarity function $sim(p, h)$ which returns scores in $[0, 1]$. While Glickman et al. used a function based on web co-occurrence statistics, we use a much simpler function based on string edit distance:

$$sim(w_1, w_2) = 1 - \frac{dist(lem(w_1), lem(w_2))}{max(|lem(w_1)|, |lem(w_2)|)}$$

(Here $lem(w)$ denotes the lemma of word w; $dist()$ denotes Levenshtein string edit distance; and $|\cdot|$ denotes string length.)

This model can be easily extended to generate an alignment score, which will be of interest in section 6. We define the score for a specific hypothesis token h to be the log of its similarity with the premise token p to which it is aligned, and the score for the complete alignment of hypothesis H to premise P to be the sum of the scores of the tokens in H, weighted by *inverse document frequency* in a large corpus[10] (so that common words get less weight), and normalized by the length of H:

$$score(h|P) = \log \max_{p \in P} sim(p, h)$$

$$score(H|P) = \frac{1}{|H|} \sum_{h \in H} idf(h) \cdot score(h|P)$$

Despite the simplicity of this alignment model, its performance is fairly robust, with good recall. Its precision, however, its mediocre—chiefly because, by design, it aligns every h with some p. The model could surely be improved by allowing it to leave some H tokens unaligned, but this was not pursued.

not greatly affected.

[10]We use $idf(w) = \log(N/N_w)$, where N is the number of documents in the corpus, and N_w is the number of documents containing word w.

System	Data	P %	R %	F_1 %	E %
Bag-of-words	dev	57.8	81.2	67.5	3.5
(baseline)	test	62.1	82.6	70.9	5.3
GIZA++	dev	83.0	66.4	72.1	9.4
(using lex, $\cap$)	test	85.1	69.1	74.8	11.3
Cross-EM	dev	67.6	80.1	72.1	1.3
(using lex, $\cap$)	test	70.3	81.0	74.1	0.8
Stanford RTE	dev	81.1	61.2	69.7	0.5
	test	82.7	61.2	70.3	0.3
Stanford RTE	dev	81.1	75.8	78.4	—
(punct. corr.)	test	82.7	75.8	79.1	—
MANLI	dev	83.4	85.5	84.4	21.7
(this work)	test	85.4	85.3	85.3	21.3

Table 1: Performance of various aligners on the MSR RTE2 alignment data. The columns show the data set used (800 problems each); average precision, recall, and F-measure; and the exact match rate (see text).

5.2 MT aligners: GIZA++ and Cross-EM

Given the importance of alignment for NLI, and the availability of standard, proven tools for MT alignment, an obvious question presents itself: why not use an off-the-shelf MT aligner for NLI? Although we have argued (section 2) that this is unlikely to succeed, to our knowledge, we are the first to investigate the matter empirically.[11]

The best-known MT aligner is undoubtedly GIZA++ (Och and Ney, 2003), which contains implementations of various IBM models (Brown et al., 1993), as well as the HMM model of Vogel et al. (1996). Most practitioners use GIZA++ as a black box, via the Moses MT toolkit (Koehn et al., 2007). We followed this practice, running with Moses' default parameters on the RTE2 data to obtain asymmetric word alignments in both directions (P-to-H and H-to-P). We then performed symmetrization using the well-known INTERSECTION heuristic.

Unsurprisingly, the out-of-the-box performance was quite poor, with most words aligned apparently at random. Precision was fair (72%) but recall was very poor (46%). Even equal words were usually not aligned—because GIZA++ is designed for cross-linguistic use, it does not consider word equality between source and target sentences. To remedy this, we supplied GIZA++ with a lexicon, using a trick

[11]However, Dolan et al. (2004) explore a closely-related topic: using an MT aligner to identify paraphrases.

common in MT: we supplemented the training data with synthetic data consisting of matched pairs of equal words. This gives GIZA++ a better chance of learning that, e.g., *man* should align with *man*. The result was a big boost in recall (+23%), and a smaller gain in precision. The results for GIZA++ shown in table 1 are based on using the lexicon and INTERSECTION. With these settings, GIZA++ properly aligned most pairs of equal words, but continued to align other words apparently at random.

Next, we compared the performance of INTERSECTION with other symmetrization heuristics defined in Moses—including UNION, GROW, GROW-DIAG, GROW-DIAG-FINAL (the default), and GROW-DIAG-FINAL-AND—and with asymmetric alignments in both directions. While all these alternatives achieved better recall than INTERSECTION, all showed substantially worse precision and F_1. On the RTE2 test set, the asymmetric alignment from H to P scored 68% in F_1; GROW scored 58%; and all other alternatives scored below 52%.

As an additional experiment, we tested the Cross-EM aligner (Liang et al., 2006) from the BerkeleyAligner package on the MSR data. While this aligner is in many ways simpler than GIZA++ (it lacks any model of fertility, for example), its method of jointly training two simple asymmetric HMM models has outperformed GIZA++ on standard evaluations of MT alignment. As with GIZA++, we experimented with a variety of symmetrization heuristics, and ran trials with and without a supplemental lexicon. The results were broadly similar: INTERSECTION greatly outperformed alternative heuristics, and using a lexicon provided a big boost (up to 12% in F_1). Under optimal settings, the Cross-EM aligner showed better recall and worse precision than GIZA++, with F_1 just slightly lower. Like GIZA++, it did well at aligning equal words, but aligned most other words at random.

The mediocre performance of MT aligners on NLI alignment comes as no surprise, for reasons discussed in section 2. Above all, the quantity of training data is simply too small for unsupervised learning to succeed. A successful NLI aligner will need to exploit supervised training data, and will need access to additional sources of knowledge about lexical relatedness.

5.3 The Stanford RTE aligner

A better comparison is thus to an alignment system expressly designed for NLI. For this purpose, we used the alignment component of the Stanford RTE system (Chambers et al., 2007). The Stanford aligner performs decoding and learning in a similar fashion to MANLI, but uses a simpler, token-based alignment representation, along with a richer set of features for alignment scoring. It represents alignments as an injective map from H tokens to P tokens. Phrase alignments are not directly representable, although the effect can be approximated by a pre-processing step which collapses multi-token named entities and certain collocations into single tokens. The features used for alignment scoring include not only measures of lexical similarity, but also syntactic features intended to promote the alignment of similar predicate-argument structures.

Despite this sophistication, the out-of-the-box performance of the Stanford aligner is mediocre, as shown in table 1. The low recall figures are particularly noteworthy. However, a partial explanation is readily available: by design, the Stanford system ignores punctuation.[12] Because punctuation tokens constitute about 15% of the aligned pairs in the MSR data, this sharply reduces measured recall. However, since punctuation matters little in inference, such recall errors probably should be forgiven. Thus, table 1 also shows adjusted statistics for the Stanford system in which all recall errors involving punctuation are (generously) ignored.

Even after this adjustment, the recall figures are unimpressive. Error analysis reveals that the Stanford aligner does a poor job of aligning function words. About 13% of the aligned pairs in the MSR data are matching prepositions or articles; the Stanford aligner misses about 67% of such pairs. (By contrast, MANLI misses only 10% of such pairs.) While function words matter less in inference than nouns and verbs, they are not irrelevant, and because sentences often contain multiple instances of a particular function word, matching them properly is by no means trivial. If matching prepositions and articles were ignored (in addition to punctuation), the gap in F_1 between the MANLI and Stanford systems

[12]In fact, it operates on a dependency-graph representation from which punctuation is omitted.

would narrow to about 2.8%.

Finally, the Stanford aligner is handicapped by its token-based alignment representation, often failing (partly or completely) to align multi-word phrases such as *peace activists* with *protesters*, or *hackers* with *non-authorized personnel*.

5.4 The MANLI aligner

As table 1 indicates, the MANLI aligner was found to outperform all other aligners evaluated on every measure of performance, achieving an F_1 score 10.5% higher than GIZA++ and 6.2% higher than the Stanford aligner (even with the punctuation correction).[13] MANLI achieved a good balance between precision and recall, and matched more than 20% of the gold-standard alignments exactly.

Three factors seem to have contributed most to MANLI's success. First, MANLI is able to outperform the MT aligners principally because it is able to leverage lexical resources to identify the similarity between pairs of words such as *jail* and *prison*, *prevent* and *stop*, or *injured* and *wounded*. Second, MANLI's contextual features enable it to do better than the Stanford aligner at matching function words, a weakness of the Stanford aligner discussed in section 5.3. Third, MANLI gains a marginal advantage because its phrase-based representation of alignment permits it to properly align phrase pairs such as *death penalty* and *capital punishment*, or *abdicate* and *give up*.

However, the phrase-based representation contributed far less than we had hoped. Setting MANLI's maximum phrase size to 1 (effectively, restricting it to token-based alignments) caused F_1 to fall by just 0.2%. We do not interpret this to mean that phrase alignments are not useful—indeed, about 2.6% of the links in the gold-standard data involve phrases of size > 1. Rather, we think it shows that we have failed to fully exploit the advantages of the phrase-based representation, chiefly because we lack lexical resources providing good information on similarity of multi-word phrases.

Error analysis suggests that there is ample room for improvement. A large proportion of recall errors (perhaps 40%) occur because the lexical similarity function assigns too low a value to pairs of words or phrases which are clearly similar, such as *conservation* and *protecting*, *server* and *computer networks*, *organization* and *agencies*, or *bone fragility* and *osteoporosis*. Better exploitation of lexical resources could help to reduce such errors. Another important category of recall errors (about 12%) result from the failure to identify one- and multi-word versions of the name of some entity, such as *Lennon* and *John Lennon*, or *Nike Inc.* and *Nike*. A special-purpose similarity function could help here. Note, however, that about 10% of recall errors are unavoidable, given our choice of alignment representation, since they involve cases where the gold standard aligns one or more tokens on one side to a non-contiguous set of tokens on the other side.

Precision errors may be harder to reduce. These errors are dominated by cases where we mistakenly align two equal function words (49% of precision errors), two forms of the verb *to be* (21%), two equal punctuation marks (7%), or two words or phrases of other types having equal lemmas (18%). Because such errors often occur because the aligner is forced to choose between nearly equivalent alternatives, they may be difficult to eliminate. The remaining 5% of precision errors result mostly from aligning words or phrases rightly judged to be highly similar, such as *expanding* and *increasing*, *labor* and *birth*, *figures* and *number*, or *223,000* and *220,000*.

6 Using alignment to predict RTE answers

In section 5, we evaluated the ability of aligners to recover gold-standard alignments. But since alignment is just one component of the NLI problem, we might also examine the impact of different aligners on the ability to recognize valid inferences. If a high-scoring alignment indicates a close correspondence between H and P, does this also indicate a valid inference? We have previously emphasized (MacCartney et al., 2006) that there is more to inferential validity than close lexical or structural correspondence: negations, modals, non-factive and implicative verbs, and other linguistic constructs can affect validity in ways hard to capture in alignment. Nevertheless, alignment score can be a strong predictor of inferential validity, and some NLI systems (e.g., (Glickman et al., 2005)) rely entirely on some measure of alignment quality to predict validity.

[13]Reported results for MANLI are averages over 10 runs.

System	data	acc %	avgP %
Bag-of-words aligner	dev	61.3	61.5
	test	57.9	58.9
Stanford RTE aligner	dev	63.1	64.9
	test	60.9	59.2
MANLI aligner	dev	59.3	69.0
(this work)	test	60.3	61.0
RTE2 entries (average)	test	58.5	59.1
LCC (Hickl et al., 2006)	test	75.4	80.8

Table 2: Performance of various aligners and complete RTE systems in predicting RTE2 answers. The columns show the data set used, accuracy, and average precision (the recommended metric for RTE2).

If an aligner generates real-valued alignment scores, we can use the RTE data to test its ability to predict inferential validity with the following simple method. For a given RTE problem, we predict YES (valid) if its alignment score[14] exceeds a threshold τ, and NO otherwise. We tune τ to maximize accuracy on the RTE2 development set, and then measure performance on the RTE2 test set using the same τ.

Table 2 shows results for several NLI aligners, along with some results for complete RTE systems, including the LCC system (the top performer at RTE2) and an average of all systems participating in RTE2. While none of the aligners rivals the performance of the LCC system, all achieve respectable results, and the Stanford and MANLI aligners outperform the average RTE2 entry. Thus, even if alignment quality does not determine inferential validity, many NLI systems could be improved by harnessing a well-designed NLI aligner.

7 Related work

Given the extensive literature on phrase-based MT, it may be helpful further to situate our phrase-based alignment model in relation to past work. The standard approach to training a phrase-based MT system is to apply phrase extraction heuristics using word-aligned training sets (Och and Ney, 2003; Koehn et al., 2007). Unfortunately, word alignment models assume that source words are individually translated into target words, which stands at odds with the key assumption in phrase-based systems that many translations are non-compositional. More recently, several works (Marcu and Wong, 2002; DeNero et al., 2006; Birch et al., 2006; DeNero and Klein, 2008) have presented more unified phrase-based systems that jointly align and weight phrases, though these systems have not come close to the state of the art when evaluated in terms of MT performance.

We would argue that previous work in MT phrase alignment is orthogonal to our work. In MANLI, the need for phrases arises when word-based representations are not appropriate for alignment (e.g., between *close down* and *terminate*), though longer phrases are not needed to achieve good alignment quality. In MT phrase alignment, it is beneficial to account for arbitrarily large phrases, since the larger contexts offered by these phrases can help realize more dependencies among translated words (e.g., word order, agreement, subcategorization). Perhaps because MT phrase alignment is dealing with much larger contexts, no existing work in MT phrase alignment (to our knowledge) directly models word insertions and deletions, as in MANLI. For example, in figure 1, MANLI can just skip *In most Pacific countries there*, while an MT phrase-based model would presumably align *In most Pacific countries there are* to *Women are*. Hence, previous work is of limited applicability to our problem.

8 Conclusion

While MT aligners succeed by unsupervised learning of word correspondences from massive amounts of bitext, NLI aligners are forced to rely on smaller quantities of supervised training data. With the MANLI system, we have demonstrated how to overcome this lack of data by utilizing external lexical resources, and how to gain additional power from a phrase-based representation of alignment.

Acknowledgements The authors wish to thank the anonymous reviewers for their helpful comments on an earlier draft of this paper. This paper is based on work funded in part by the Defense Advanced Research Projects Agency through IBM and in part by the CIA ATP as part of the OCCAM project.

[14]For good results, it may be necessary to normalize the alignment score. Scores from MANLI were normalized by the number of tokens in the problem. The Stanford aligner performs a similar normalization internally.

References

R. Bar-Haim, I. Dagan, B. Dolan, L. Ferro, D. Giampiccolo, B. Magnini, and I. Szpektor. 2006. The Second PASCAL Recognising Textual Entailment Challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

R. Bar-Haim, I. Dagan, I. Greental, and E. Shnarch. 2007. Semantic Inference at the Lexical-Syntactic Level. In *Proceedings of AAAI-07*.

A. Birch, C. Callison-Burch, M. Osborne, and P. Koehn. 2006. Constraining the Phrase-Based, Joint Probability Statistical Translation Model. In *Proceedings of the ACL-06 Workshop on Statistical Machine Translation*.

C. Brockett. 2007. Aligning the RTE 2006 Corpus. Technical Report MSR-TR-2007-77, Microsoft Research.

P. F. Brown, S. D. Pietra, V. J. D. Pietra, and R. L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311.

N. Chambers, D. Cer, T. Grenager, D. Hall, C. Kiddon, B. MacCartney, M. C. de Marneffe, D. Ramage, E. Yeh, and C. D. Manning. 2007. Learning Alignments and Leveraging Natural Logic. In *Proceedings of the ACL-07 Workshop on Textual Entailment and Paraphrasing*.

M. Collins. 2002. Discriminative training methods for hidden Markov models. In *Proceedings of EMNLP-02*.

I. Dagan, O. Glickman, and B. Magnini. 2005. The PASCAL Recognising Textual Entailment Challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*.

J. DeNero and D. Klein. 2008. The Complexity of Phrase Alignment Problems. In *Proceedings of ACL/HLT-08: Short Papers*, pages 25–28.

J. DeNero, D. Gillick, J. Zhang, and D. Klein. 2006. Why Generative Phrase Models Underperform Surface Heuristics. In *Proceedings of the ACL-06 Workshop on Statistical Machine Translation*, pages 31–38.

B. Dolan, C. Quirk, and C. Brockett. 2004. Unsupervised construction of large paraphrase corpora. In *Proceedings of COLING-04*.

A. Fraser and D. Marcu. 2007. Measuring Word Alignment Quality for Statistical Machine Translation. *Computational Linguistics*, 33(3):293–303.

O. Glickman, I. Dagan, and M. Koppel. 2005. Web based probabilistic textual entailment. In *Proceedings of the PASCAL Challenges Workshop on Recognizing Textual Entailment*.

A. Hickl and J. Bensley. 2007. A Discourse Commitment-Based Framework for Recognizing Textual Entailment. In *ACL-07 Workshop on Textual Entailment and Paraphrasing*, Prague.

A. Hickl, J. Williams, J. Bensley, K. Roberts, B. Rink, and Y. Shi. 2006. Recognizing Textual Entailment with LCC's GROUNDHOG System. In *Proceedings of the Second PASCAL Challenges Workshop on Recognizing Textual Entailment*.

J. J. Jiang and D. W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the International Conference on Research in Computational Linguistics*.

V. Jijkoun and M. de Rijke. 2005. Recognizing textual entailment using lexical similarity. In *Proceedings of the PASCAL Challenges Workshop on Recognizing Textual Entailment*, pages 73–76.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL-07, demonstration session*.

P. Liang, B. Taskar, and D. Klein. 2006. Alignment by Agreement. In *Proceedings of NAACL-06*, New York.

D. Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING/ACL-98*, pages 768–774, Montreal, Canada.

B. MacCartney, T. Grenager, M. C. de Marneffe, D. Cer, and C. D. Manning. 2006. Learning to Recognize Features of Valid Textual Entailments. In *Proceedings of NAACL-06*, New York.

D. Marcu and W. Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of EMNLP-02*, pages 133–139.

E. Marsi and E. Krahmer. 2005. Classification of semantic relations by humans and machines. In *ACL-05 Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, Ann Arbor.

F. J. Och and H. Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51.

M. Tatu and D. Moldovan. 2007. COGEX at RTE3. In *Proceedings of ACL-07*.

S. Vogel, H. Ney, and C. Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of COLING-96*, pages 836–841, Copenhagen, Denmark.

Attacking Decipherment Problems Optimally with Low-Order N-gram Models

Sujith Ravi and Kevin Knight
University of Southern California
Information Sciences Institute
Marina del Rey, California 90292
{sravi,knight}@isi.edu

Abstract

We introduce a method for solving substitution ciphers using low-order letter n-gram models. This method enforces global constraints using integer programming, and it guarantees that no decipherment key is overlooked. We carry out extensive empirical experiments showing how decipherment accuracy varies as a function of cipher length and n-gram order. We also make an empirical investigation of Shannon's (1949) theory of uncertainty in decipherment.

1 Introduction

A number of papers have explored algorithms for automatically solving letter-substitution ciphers. Some use heuristic methods to search for the best deterministic key (Peleg and Rosenfeld, 1979; Ganesan and Sherman, 1993; Jakobsen, 1995; Olson, 2007), often using word dictionaries to guide that search. Others use expectation-maximization (EM) to search for the best probabilistic key using letter n-gram models (Knight et al., 2006). In this paper, we introduce an exact decipherment method based on integer programming. We carry out extensive decipherment experiments using letter n-gram models, and we find that our accuracy rates far exceed those of EM-based methods.

We also empirically explore the concepts in Shannon's (1949) paper on information theory as applied to cipher systems. We provide quantitative plots for uncertainty in decipherment, including the famous *unicity distance*, which estimates how long a cipher must be to virtually eliminate such uncertainty.

We find the ideas in Shannon's (1949) paper relevant to problems of statistical machine translation and transliteration. When first exposed to the idea of statistical machine translation, many people naturally ask: (1) how much data is needed to get a good result, and (2) can translation systems be trained without parallel data? These are tough questions by any stretch, and it is remarkable that Shannon was already in the 1940s tackling such questions in the realm of code-breaking, creating analytic formulas to estimate answers.

Our novel contributions are as follows:

- We outline an exact letter-substitution decipherment method which:
  - guarantees that no key is overlooked, and
  - can be executed with standard integer programming solvers
- We present empirical results for decipherment which:
  - plot search-error-free decipherment results at various cipher lengths, and
  - demonstrate accuracy rates superior to EM-based methods
- We carry out empirical testing of Shannon's formulas for decipherment uncertainty

2 Language Models

We work on letter substitution ciphers with spaces. We look for the key (among 26! possible ones) that, when applied to the ciphertext, yields the most English-like result. We take "English-like" to mean most probable according to some statistical language model, whose job is to assign some probability to any sequence of letters. According to a 1-gram model of English, the probability of a plaintext $p_1 ... p_n$ is given by:

$$P(p_1...p_n) = P(p_1) \cdot P(p_2) \cdot ... \cdot P(p_n)$$

That is, we obtain the probability of a sequence by multiplying together the probabilities of the individual letters that make it up. This model assigns a probability to any letter sequence, and the probabilities of all letter sequences sum to one. We collect letter probabilities (including space) from 50 million words of text available from the Linguistic Data Consortium (Graff and Finch, 1994). We also estimate 2- and 3-gram models using the same resources:

$$\begin{aligned} P(p_1...p_n) &= P(p_1 \mid START) \cdot P(p_2 \mid p_1) \cdot P(p_3 \mid p_2) \cdot \\ &\quad ... \cdot P(p_n \mid p_{n-1}) \cdot P(END \mid p_n) \end{aligned}$$

$$\begin{aligned} P(p_1...p_n) &= P(p_1 \mid START) \cdot P(p_2 \mid START\, p_1) \cdot \\ &\quad P(p_3 \mid p_1\, p_2) \cdot ... \cdot P(p_n \mid p_{n-2}\, p_{n-1}) \cdot \\ &\quad P(END \mid p_{n-1}\, p_n) \end{aligned}$$

Unlike the 1-gram model, the 2-gram model will assign a low probability to the sequence "ae" because the probability $P(e \mid a)$ is low. Of course, all these models are fairly weak, as already known by (Shannon, 1949). When we stochastically generate text according to these models, we get, for example:

1-gram: ... thdo detusar ii c ibt deg irn toihytrsen ...
2-gram: ... itariaris s oriorcupunond rke uth ...
3-gram: ... ind thnowelf jusision thad inat of ...
4-gram: ... rece bence on but ther servier ...
5-gram: ... mrs earned age im on d the perious ...
6-gram: ... a party to possible upon rest of ...
7-gram: ... t our general through approve the ...

We can further estimate the probability of a whole English sentence or phrase. For example, the probabilities of two plaintext phrases "het oxf" and "the fox" (which have the same letter frequency distribution) is shown below. The 1-gram model which counts only the frequency of occurrence of each letter in the phrase, estimates the same probability for both the phrases "het oxf" and "the fox", since the same letters occur in both phrases. On the other hand, the 2-gram and 3-gram models, which take context into account, are able to distinguish between the English and non-English phrases better, and hence assign a higher probability to the English phrase "the fox".

Model	P(het oxf)	P(the fox)
1-gram	1.83×10^{-9}	1.83×10^{-9}
2-gram	3.26×10^{-11}	1.18×10^{-7}
3-gram	1.89×10^{-13}	1.04×10^{-6}

Over a longer sequence X of length N, we can also calculate $-log_2(P(X))/N$, which (per Shannon) gives the compression rate permitted by the model, in bits per character. In our case, we get:[1]

1-gram: 4.19
2-gram: 3.51
3-gram: 2.93

3 Decipherment

Given a ciphertext $c_1...c_n$, we search for the key that yields the most probable plaintext $p_1...p_n$. There are 26! possible keys, too many to enumerate. However, we can still find the best one in a guaranteed fashion. We do this by taking our most-probable-plaintext problem and casting it as an integer programming problem.[2]

Here is a sample integer programming problem:

variables: x, y
minimize:
$$2x + y$$
subject to:
$$x + y < 6.9$$
$$y - x < 2.5$$
$$y > 1.1$$

We require that x and y take on integer values. A solution can be obtained by typing this integer program into the publicly available lp_solve program,

[1]Because spacing is fixed in our letter substitution ciphers, we normalize $P(X)$ by the sum of probabilities of all English strings that match the spacing pattern of X.

[2]For an overview of integer and linear programming, see for example (Schrijver, 1998).

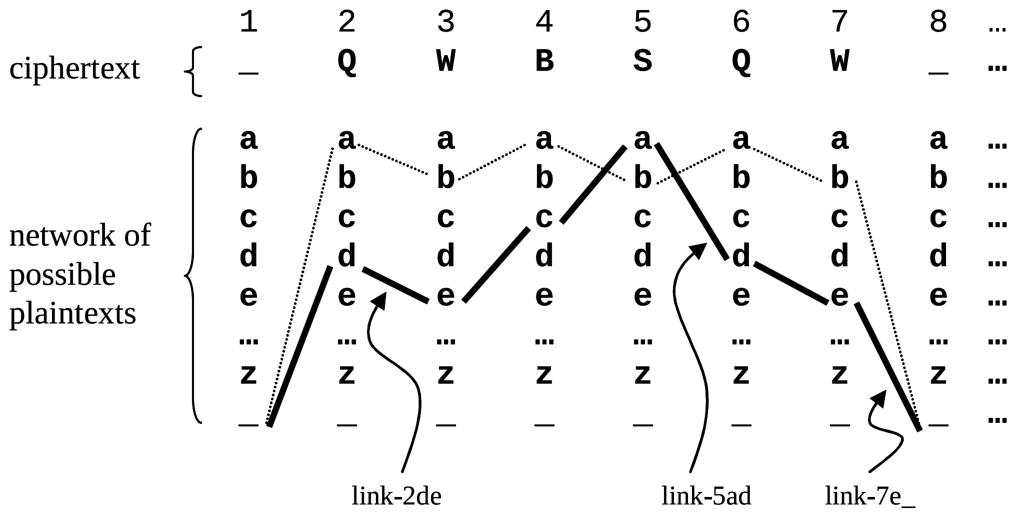


Figure 1: A decipherment network. The beginning of the ciphertext is shown at the top of the figure (underscores represent spaces). Any left-to-right path through the network constitutes a potential decipherment. The bold path corresponds to the decipherment "decade". The dotted path corresponds to the decipherment "ababab". Given a cipher length of n, the network has $27 \cdot 27 \cdot (n-1)$ links and 27^n paths. Each link corresponds to a named variable in our integer program. Three links are shown with their names in the figure.

or the commercially available $CPLEX$ program, which yields the result: $x = 4, y = 2$.

Suppose we want to decipher with a 2-gram language model, i.e., we want to find the key that yields the plaintext of highest 2-gram probability. Given the ciphertext $c_1...c_n$, we create an integer programming problem as follows. First, we set up a network of possible decipherments (Figure 1). Each of the $27 \cdot 27 \cdot (n-1)$ links in the network is a binary variable in the integer program—it must be assigned a value of either 0 or 1. We name these variables $link_{XYZ}$, where X indicates the column of the link's source, and Y and Z represent the rows of the link's source and destination (e.g. variables $link_{1aa}$, $link_{1ab}$, $link_{5qu}$, ...).

Each distinct left-to-right path through the network corresponds to a different decipherment. For example, the bold path in Figure 1 corresponds to the decipherment "decade". Decipherment amounts to turning some links "on" (assigning value 1 to the link variable) and others "off" (assigning value 0). Not all assignments of 0's and 1's to link variables result in a coherent left-to-right path, so we must place some "subject to" constraints in our integer program.

We observe that a set of variables forms a path if, for every node in columns 2 through $n-1$ of the network, the following property holds: the sum of the values of the link variables entering the node equals the sum of the link variables leaving the node. For nodes along a chosen decipherment path, this sum will be 1, and for others, it will be 0.[3] Therefore, we create one "subject to" constraint for each node ("_" stands for space). For example, for the node in column 2, row e we have:

subject to:

$$link_{1ae} + link_{1be} + link_{1ce} + ... + link_{1\_e}$$
$$= link_{2ea} + link_{2eb} + link_{2ec} + ... + link_{2e\_}$$

Now we set up an expression for the "minimize" part of the integer program. Recall that we want to select the plaintext $p_1...p_n$ of highest probability. For the 2-gram language model, the following are equivalent:

(a) Maximize $P(p_1...p_n)$

(b) Maximize $log_2\, P(p_1...p_n)$

(c) Minimize $-log_2\, P(p_1...p_n)$

(d) Minimize $-log_2\, [\ P(p_1 \mid START)$

[3]Strictly speaking, this constraint over nodes still allows multiple decipherment paths to be active, but we can rely on the rest of our integer program to select only one.

$$\begin{aligned} &\cdot P(p_2 \,|\, p_1) \\ &\cdot \;... \\ &\cdot P(p_n \,|\, p_{n-1}) \\ &\cdot P(END \,|\, p_n)\,] \end{aligned}$$

(e) Minimize
$$\begin{aligned} &-log_2\, P(p_1 \,|\, START) \\ &-log_2\, P(p_2 \,|\, p_1) \\ &-\; ... \\ &-log_2\, P(p_n \,|\, p_{n-1}) \\ &-log_2\, P(END \,|\, p_n) \end{aligned}$$

We can guarantee this last outcome if we construct our minimization function as a sum of $27 \cdot 27 \cdot (n-1)$ terms, each of which is a $link_{XYZ}$ variable multiplied by $-log_2 P(Z|Y)$:

Minimize
$$\begin{aligned} & link_{1aa} \cdot -log_2\, P(a \mid a) \\ &+ link_{1ab} \cdot -log_2\, P(b \mid a) \\ &+ link_{1ac} \cdot -log_2\, P(c \mid a) \\ &+ ... \\ &+ link_{5qu} \cdot -log_2\, P(u \mid q) \\ &+ ... \end{aligned}$$

When we assign value 1 to link variables along some decipherment path, and 0 to all others, this function computes the negative log probability of that path.

We must still add a few more "subject to" constraints. We need to ensure that the chosen path imitates the repetition pattern of the ciphertext. While the bold path in Figure 1 represents the fine plaintext choice "decade", the dotted path represents the choice "ababab", which is not consistent with the repetition pattern of the cipher "QWBSQW". To make sure our substitutions obey a consistent key, we set up $27 \cdot 27 = 729$ new key_{xy} variables to represent the choice of key. These new variables are also binary, taking on values 0 or 1. If variable key_{aQ} = 1, that means the key maps plaintext a to ciphertext Q. Clearly, not all assignments to these 729 variables represent valid keys, so we augment the "subject to" part of our integer program by requiring that for any letter x,

subject to:

$$\begin{aligned} & key_{xA} + key_{xB} + ... + key_{xZ} + key_{x\_} = 1 \\ & key_{Ax} + key_{Bx} + ... + key_{Zx} + key_{\_x} = 1 \end{aligned}$$

That is, every plaintext letter must map to exactly one ciphertext letter, and every ciphertext letter must map to exactly one plaintext letter. We also add a constraint to ensure that the ciphertext space character maps to the plaintext space character:

subject to:

$$key_{\_\_} = 1$$

Finally, we ensure that any chosen decipherment path of $link_{XYZ}$ variables is consistent with the chosen key. We know that for every node A along the decipherment path, exactly one active link has A as its destination. For all other nodes, zero active links lead in. Suppose node A represents the decipherment of ciphertext letter c_i as plaintext letter p_j—for all such nodes, we stipulate that the sum of values for $link_{(i-1)xp_j}$ (for all x) equals the value of $key_{p_j c_i}$. In other words, whether a node lies along the chosen decipherment path or not, the chosen key must support that decision.

Figure 2 summarizes the integer program that we construct from a given ciphertext $c_1...c_n$. The computer code that transforms any given cipher into a corresponding integer program runs to about one page. Variations on the decipherment network yield 1-gram and 3-gram decipherment capabilities. Once an integer program is generated by machine, we ask the commercially-available $CPLEX$ software to solve it, and then we note which key_{XY} variables are assigned value 1. Because $CPLEX$ computes the optimal key, the method is not fast—for ciphers of length 32, the number of variables and constraints encoded in the integer program (IP) along with average running times are shown below. It is possible to obtain less-than-optimal keys faster by interrupting the solver.

Model	**# of IP variables**	**# of IP constraints**	**Average running time**
1-gram	$1,755$	$1,083$	0.01 seconds
2-gram	$27,700$	$2,054$	50 seconds
3-gram	$211,600$	$27,326$	450 seconds

4 Decipherment Experiments

We create 50 ciphers each of lengths $2, 4, 8, ..., 256$. We solve these with 1-gram, 2-gram, and 3-gram language models. We record the average percentage of ciphertext tokens decoded incorrectly. 50% error means half of the ciphertext tokens are deciphered wrong, while 0% means perfect decipherment. Here

variables:

$link_{ipr}$ 1 if the *i*th cipher letter is deciphered as plaintext letter *p* AND the *(i+1)*th cipher letter is deciphered as plaintext letter *r*
0 otherwise

key_{pq} 1 if decipherment key maps plaintext letter *p* to ciphertext letter *q*
0 otherwise

minimize:

$\sum_{i=1}^{n-1} \sum_{p,r} link_{ipr} \cdot -log\ P(r|p)$ (2-gram probability of chosen plaintext)

subject to:

for all *p*: $\sum_r key_{pr} = 1$ (each plaintext letter maps to exactly one ciphertext letter)

for all *p*: $\sum_r key_{rp} = 1$ (each ciphertext letter maps to exactly one plaintext letter)

$key_{\_\ \_} = 1$ (cipher space character maps to plain space character)

for (*i=1...n-2*), for all *r*: $[\sum_p link_{ipr} = \sum_p link_{(i+1)rp}]$ (chosen links form a left-to-right path)

for (*i=1...n-1*), for all *p*: $\sum_r link_{irp} = key_{pc_{i+1}}$ (chosen links are consistent with chosen key)

Figure 2: Summary of how to build an integer program for any given ciphertext $c_1...c_n$. Solving the integer program will yield the decipherment of highest probability.

we illustrate some automatic decipherments with error rates:

42% error: the avelage ongrichman hal cy wiof a sevesonme qus antizexty that he buprk lathes we blung than soment - fotes mmasthes

11% error: the average englishman has so week a reference for antiality that he would rather be prong than recent - deter quarteur

2% error: the average englishman has so keep a reference for antiquity that he would rather be wrong than recent - peter mcarthur

0% error: the average englishman has so deep a reverence for antiquity that he would rather be wrong than recent - peter mcarthur

Figure 3 shows our automatic decipherment results. We note that the solution method is exact, not heuristic, so that decipherment error is not due to search error. Our use of global key constraints also leads to accuracy that is superior to (Knight et al., 2006). With a 2-gram model, their EM algorithm gives 10% error for a 414-letter cipher, while our method provides a solution with only 0.5% error. At shorter cipher lengths, we observe much higher improvements when using our method. For example, on a 52-letter textbook cipher, using a 2-gram model, the solution from our method resulted in 21% error as compared to 85% error given by the EM solution.

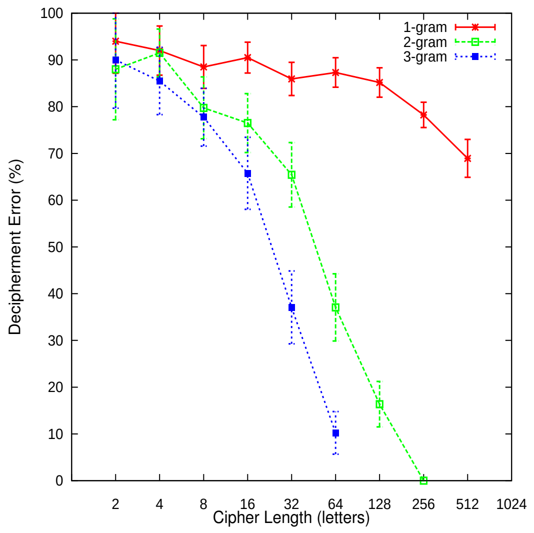


Figure 3: Average decipherment error using integer programming vs. cipher length, for 1-gram, 2-gram and 3-gram models of English. Error bars indicate 95% confidence intervals.

We see that deciphering with 3-grams works well on ciphers of length 64 or more. This confirms that such ciphers can be attacked with very limited knowledge of English (no words or grammar) and little custom programming.

The 1-gram model works badly in this scenario, which is consistent with Bauer's (2006) observation that for short texts, mechanical decryption on the basis of individual letter frequencies does not work. If we had infinite amounts of ciphertext and plaintext drawn from the same stochastic source, we would expect the plain and cipher frequencies to eventually line up, allowing us to read off a correct key from the frequency tables. The upper curve in Figure 3 shows that convergence to this end is slow.

5 Shannon Equivocation and Unicity Distance

Very short ciphers are hard to solve accurately. Shannon (1949) pinpointed an inherent difficulty with short ciphers, one that is independent of the solution method or language model used; the cipher itself may not contain enough information for its proper solution. For example, given a short cipher like $XYYX$, we can never be sure if the answer is *peep*, *noon*, *anna*, etc. Shannon defined a mathematical measure of our decipherment uncertainty, which he called *equivocation* (now called entropy).

Let C be a cipher, M be the plaintext message it encodes, and K be the key by which the encoding takes place. Before even seeing C, we can compute our uncertainty about the key K by noting that there are 26! equiprobable keys:[4]

$$\begin{aligned} H(K) &= -(26!) \cdot (1/26!) \cdot log_2\,(1/26!) \\ &= 88.4 \text{ bits} \end{aligned}$$

That is, any secret key can be revealed in 89 bits. When we actually receive a cipher C, our uncertainty about the key and the plaintext message is reduced. Shannon described our uncertainty about the plaintext message, letting m range over all decipherments:

$$\begin{aligned} H(M|C) &= \text{equivocation of plaintext message} \\ &= -\sum_m P(m|C) \cdot log_2\, P(m|C) \end{aligned}$$

[4](Shannon, 1948) The entropy associated with a set of possible events whose probabilities of occurrence are $p_1, p_2, ..., p_n$ is given by $H = -\sum_{i=1}^{n} p_i \cdot log_2(p_i)$.

$P(m|C)$ is probability of plaintext m (according to the language model) divided by the sum of probabilities of all plaintext messages that obey the repetition pattern of C. While integer programming gives us a method to find the most probable decipherment without enumerating all keys, we do not know of a similar method to compute a full equivocation without enumerating all keys. Therefore, we sample up to 100,000 plaintext messages in the neighborhood of the most probably decipherment[5] and compute $H(M|C)$ over that subset.[6]

Shannon also described $H(K|C)$, the *equivocation of key*. This uncertainty is typically larger than $H(M|C)$, because a given message M may be derived from C via more than one key, in case C does not contain all 26 letters of the alphabet.

We compute $H(K|C)$ by letting $r(C)$ be the number of distinct letters in C, and letting $q(C)$ be $(26 - r(C))!$. Letting i range over our sample of plaintext messages, we get:

$$\begin{aligned} H(K|C) &= \text{equivocation of key} \\ &= -\sum_i q(C) \cdot (P(i)/q(C)) \cdot log_2\,(P(i)/q(C)) \\ &= -\sum_i P(i) \cdot log_2\,(P(i)/q(C)) \\ &= -\sum_i P(i) \cdot (log_2\, P(i) - log_2\, q(C)) \\ &= -\sum_i P(i) \cdot log_2\, P(i) + \sum_i P(i) \cdot log_2\, q(C) \\ &= H(M|C) + log_2\, q(C) \end{aligned}$$

Shannon (1949) used analytic means to roughly sketch the curves for $H(K|C)$ and $H(M|C)$, which we reproduce in Figure 4. Shannon's curve is drawn for a human-level language model, and the y-axis is given in "decimal digits" instead of bits.

[5]The sampling used to compute $H(M|C)$ starts with the optimal key and expands out a frontier, by swapping letters in the key, and recursing to generate new keys (and corresponding plaintext message decipherments). The plaintext messages are remembered so that the frontier expands efficiently. The sampling stops if 100,000 different messages are found.

[6]Interestingly, as we grow our sample out from the most probable plaintext, we do not guarantee that any intermediate result is a lower bound on the equivocation. An example is provided by the growing sample (0.12, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01), whose entropy steadily increases. However, if we add a 14th item whose $P(m)$ is 0.12, the entropy suddenly decreases from 2.79 to 2.78.

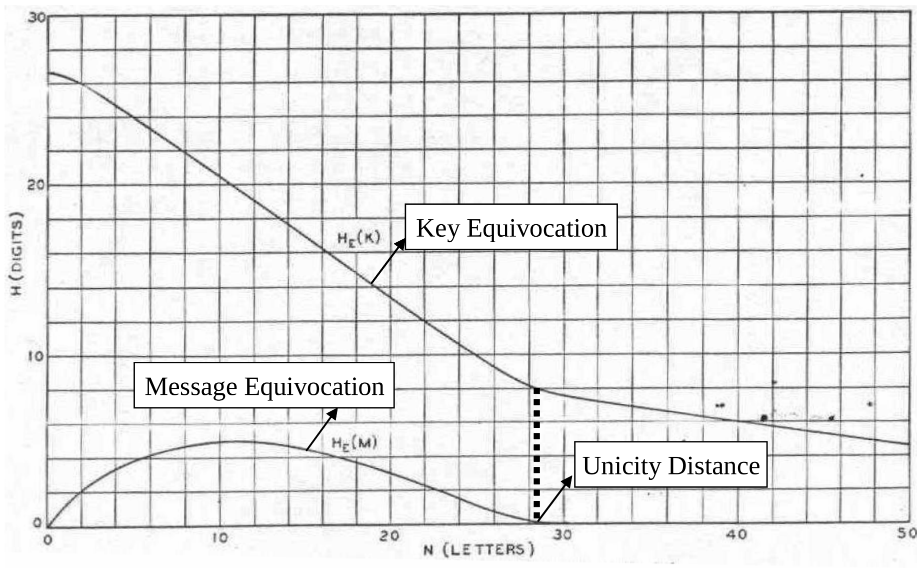


Figure 4: Equivocation for simple substitution on English (Shannon, 1949).

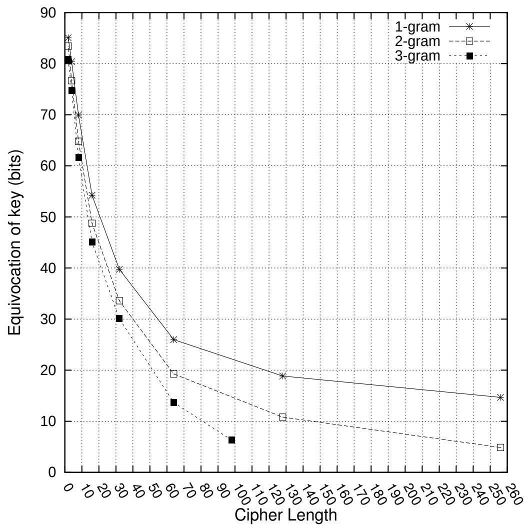


Figure 5: Average key equivocation observed (bits) vs. cipher length (letters), for 1-gram, 2-gram and 3-gram models of English.

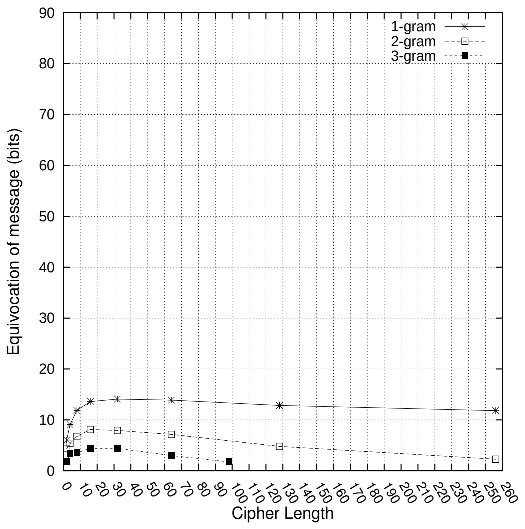


Figure 6: Average message equivocation observed (bits) vs. cipher length (letters), for 1-gram, 2-gram and 3-gram models of English.

For comparison, we plot in Figures 5 and 6 the average equivocations as we empirically observe them using our 1-, 2-, and 3-gram language models.

The shape of the key equivocation curve follows Shannon, except that it is curved from the start, rather than straight.

The message equivocation curve follows Shannon's prediction, rising then falling. Because very short ciphers have relatively few solutions (for example, a one-letter cipher has only 26), the overall uncertainty is not that great.[7] As the cipher gets longer, message equivocation rises. At some point, it then decreases, as the cipher begins to reveal its secret through patterns of repetition.

Shannon's analytic model also predicts a sharp decline of message equivocation towards zero. He

[7]Uncertainty is only loosely related to accuracy—even if we are quite certain about a solution, it may still be wrong.

defines the *unicity distance* (U) as the cipher length at which we have virtually no more uncertainty about the plaintext. Using analytic means (and various approximations), he gives:

$$U = H(K)/(A - B)$$

where:

A	=	bits per character of a 0-gram model (4.7)
B	=	bits per character of the model used to decipher

For a human-level language model ($B \sim 1.2$), he concludes $U \sim 25$, which is confirmed by practice. For our language models, the formula gives:

$$\begin{aligned} U &= 173 \text{ (1-gram)} \\ U &= 74 \text{ (2-gram)} \\ U &= 50 \text{ (3-gram)} \end{aligned}$$

These numbers are in the same ballpark as Bauer (2006), who gives 167, 74, and 59. We note that these predicted unicity distances are a bit too rosy, according to our empirical message equivocation curves. Our experience confirms this as well, as 1-gram frequency counts over a 173-letter cipher are generally insufficient to pin down a solution.

6 Conclusion

We provide a method for deciphering letter substitution ciphers with low-order models of English. This method, based on integer programming, requires very little coding and can perform an optimal search over the key space. We conclude by noting that English language models currently used in speech recognition (Chelba and Jelinek, 1999) and automated language translation (Brants et al., 2007) are much more powerful, employing, for example, 7-gram word models (not letter models) trained on trillions of words. Obtaining optimal keys according to such models will permit the automatic decipherment of shorter ciphers, but this requires more specialized search than what is provided by general integer programming solvers. Methods such as these should also be useful for natural language decipherment problems such as character code conversion, phonetic decipherment, and word substitution ciphers with applications in machine translation (Knight et al., 2006).

7 Acknowledgements

The authors wish to gratefully acknowledge Jonathan Graehl, for providing a proof to support the argument that taking a larger number of samples does not necessarily increase the equivocation. This research was supported by the Defense Advanced Research Projects Agency under SRI International's prime Contract Number NBCHD040058.

References

Friedrich L. Bauer. 2006. *Decrypted Secrets: Methods and Maxims of Cryptology*. Springer-Verlag.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of EMNLP-CoNLL*.

Ciprian Chelba and Frederick Jelinek. 1999. Structured language modeling for speech recognition. In *Proceedings of NLDB: 4th International Conference on Applications of Natural Language to Information Systems*.

Ravi Ganesan and Alan T. Sherman. 1993. Statistical techniques for language recognition: An introduction and guide for cryptanalysts. *Cryptologia*, 17(4):321–366.

David Graff and Rebecca Finch. 1994. Multilingual text resources at the linguistic data consortium. In *Proceedings of the HLT Workshop on Human Language Technology*.

Thomas Jakobsen. 1995. A fast method for cryptanalysis of substitution ciphers. *Cryptologia*, 19(3):265–274.

Kevin Knight, Anish Nair, Nishit Rathod, and Kenji Yamada. 2006. Unsupervised analysis for decipherment problems. In *Proceedings of the COLING/ACL*.

Edwin Olson. 2007. Robust dictionary attack of short simple substitution ciphers. *Cryptologia*, 31(4):332–342.

Shmuel Peleg and Azriel Rosenfeld. 1979. Breaking substitution ciphers using a relaxation algorithm. *Comm. ACM*, 22(11):598–605.

Alexander Schrijver. 1998. *Theory of Linear and Integer Programming*. John Wiley & Sons.

Claude E. Shannon. 1948. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656.

Claude E. Shannon. 1949. Communication theory of secrecy systems. *Bell System Technical Journal*, 28:656–715.

Integrating Multi-level Linguistic Knowledge with a Unified Framework for Mandarin Speech Recognition

Xinhao Wang, Jiazhong Nie, Dingsheng Luo, Xihong Wu[*]
Speech and Hearing Research Center,
Key Laboratory of Machine Perception (Ministry of Education),
School of Electronics Engineering and Computer Science,
Peking University, Beijing, 100871, China
{wangxh,niejz,wxh,dsluo}@cis.pku.edu.cn

Abstract

To improve the Mandarin large vocabulary continuous speech recognition (LVCSR), a unified framework based approach is introduced to exploit multi-level linguistic knowledge. In this framework, each knowledge source is represented by a Weighted Finite State Transducer (WFST), and then they are combined to obtain a so-called analyzer for integrating multi-level knowledge sources. Due to the uniform transducer representation, any knowledge source can be easily integrated into the analyzer, as long as it can be encoded into WFSTs. Moreover, as the knowledge in each level is modeled independently and the combination is processed in the model level, the information inherently in each knowledge source has a chance to be thoroughly exploited. By simulations, the effectiveness of the analyzer is investigated, and then a LVCSR system embedding the presented analyzer is evaluated. Experimental results reveal that this unified framework is an effective approach which significantly improves the performance of speech recognition with a 9.9% relative reduction of character error rate on the HUB-4 test set, a widely used Mandarin speech recognition task.

1 Introduction

Language modeling is essential for large vocabulary continuous speech recognition (LVCSR), which aims to determine the prior probability of a supposed word string W, $p(W)$. Although the word-based n-gram language model remains the mainstream for most speech recognition systems, the utilization of linguistic knowledge is too limited in this model. Consequently, many researchers have focused on introducing more linguistic knowledge in language modeling, such as lexical knowledge , syntax and semantics of language (Wang and Vergyri, 2006; Wang et al., 2004; Charniak, 2001; Roark, 2001; Chelba, 2000; Heeman, 1998; Chelba et al., 1997).

Recently, structured language models have been introduced to make use of syntactic hierarchical characteristics (Roark, 2001; Charniak, 2001; Chelba, 2000). Nevertheless, the computational complexity of decoding will be heavily increased, as they are parser-based models. In contrast, the class-based language model groups the words that have similar functions of syntax or semantics into meaningful classes. As a result, it handles the questions of data sparsity and generalization of unseen event. In practice, the part-of-speech (POS) information, capturing the syntactic role of words, has been widely used in clustering words (Wang and Vergyri, 2006; Maltese et al., 2001; Samuelsson and Reichl, 1999). In Heeman's POS language model (Heeman, 1998), the joint probability of word sequence and associated POS sequence was estimated directly, which has been demonstrated to be superior to the conditional probability previously used in the class-based models (Johnson, 2001). Moreover, a SuperARV language model was presented (Wang and Harper, 2002), in which lexical features and syntactic constraints were tightly integrated into a linguistic structure of SuperARV serving as a class in the model. Thus, these knowledge was integrated in the representation level, and then the joint probabilities

[*]Corresponding author: Xihong Wu

of words and corresponding SuperARVs were estimated. However, in the class-based language models, words are taken as the model units, while other units smaller or larger than words are unfeasible for modeling simultaneously, such as the Chinese characters for Chinese names.

Usually, speech recognition systems can only recognize the words within a predefined dictionary. With the increase of unknown words, i.e., out-of-vocabulary (OOV) words, the performance will degrade dramatically. This is because not only those unknown words cannot be recognized correctly, but the words surrounding them will be affected. Thus, many efforts have been made to deal with the issue of OOV words (Martins et al., 2006; Galescu, 2003; Bazzi and Glass, 2001), and various model units smaller than words have been examined to recognize OOVs from speech, such as phonemes (Bazzi and Glass, 2000a), variable-length phoneme sequence (Bazzi and Glass, 2001), syllable (Bazzi and Glass, 2000b) and sub-word (Galescu, 2003). Since the proper name is a typical category of OOV words and usually takes a very large proportion among all kinds of OOV words, it has been specially addressed in (Hu et al., 2006; Tanigaki et al., 2000).

All those attempts mentioned above succeed in utilizing linguistic knowledge in language modeling in some degree respectively. In this study, a unified framework based approach, which aims to exploit information from multi-level linguistic knowledge, is presented. Here, the Weighted Finite State Transducer (WFST) turns to be an ideal choice for our purpose. WFSTs were formerly introduced to simplify the integration of models in speech recognition, including acoustic models, phonetic models and word n-gram (Mohri, 1997; Mohri et al., 2002). In recent years, the WFST has been successfully applied in several state-of-the-art speech recognition systems, such as systems developed by the AMI project (Hain et al., 2006), IBM (Saon et al., 2003) and AT&T (Mohri et al., 1996), and in various fields of natural language processing, such as smoothed n-gram model, partial parsing (Abney, 1996), named entities recognition (Friburger and Maurel, 2004), semantic interpretation (Raymond et al., 2006) and machine translation (Tsukada and Nagata, 2004). In (Takaaki Hori and Minami, 2003), the WFST has been further used for language model adaptation, where language models of different vocabularies that represented different styles were integrated through the framework of speech translation. In WFST-based systems, all of the models are represented uniformly by WFSTs, and the general composition algorithm (Mohri et al., 2000) combines these representations flexibly and efficiently. Thereby, rather than integrating the models step by step in decoding stage, a complete search network is constructed in advance. The combined WFST will be more efficient by optimizing with determinization, minimization and pushing algorithms of WFSTs (Mohri, 1997). Besides, the researches on optimizing the search space and improving WFST-based speech recognition has been carried out, especially on how to perform on-the-fly WFSTs composition more efficiently (Hori et al., 2007; Diamantino Caseiro, 2002).

In this study, we extend the linguistic knowledge used in speech recognition. As WFSTs provide a common and natural representation for lexical constraints, n-gram language model, Hidden Markov Model models and context-dependency, multi-level knowledge sources can be encoded into WFSTs under the uniform transducer representation. Then this group of WFSTs is flexibly combined together to obtain an analyzer representing knowledge of person and location names as well as POS information. Afterwards, the presented analyzer is incorporated into LVCSR to evaluate the linguistic correctness of recognition candidates by an n-best rescoring.

Unlike other methods, this approach holds two distinct features. Firstly, as all multi-level knowledge sources are modeled independently, the model units such as character, words, phrase, etc., can be chosen freely. Meanwhile, the integration of these information sources is conducted in the model level rather than the representation level. This setup will help to model each knowledge source sufficiently and may promote the accuracy of speech recognition. Secondly, under this unified framework, it is easy to combine additional knowledge source into the framework with the only requirement that the new knowledge source can be represented by WFSTs. Moreover, since all knowledge sources are finally represented by a single WFST, additional efforts are not required for decoding the new knowledge source.

The remainder of this paper is structured as follows. In section 2, we introduce our analyzer in detail, and incorporate it into a Mandarin speech recognition system. In section 3, the simulations are performed to evaluate the analyzer and test its effectiveness when being applied to LVCSR. The conclusion appears in section 4.

2 Incorporation of Multi-level linguistic knowledge in LVCSR

In this section, we start by giving a brief description on WFSTs. Then some special characteristics of Chinese are investigated, and the model units are fixed. Afterwards, each knowledge source is represented with WFSTs, and then they are combined into a final WFST, so-called analyzer. At last, this analyzer is incorporated into Mandarin LVCSR.

2.1 Weighted Finite State Transducers

The Weighted Finite State Transducer (WFST) is the generalization of the finite state automata, in which, besides of an input label, an output label and a weight are also placed on each transition. With these labels, a WFST is capable of realizing a weighted relation between strings. In our system, log probabilities are adopted as transition weights and the relation between two strings is associated with a weight indicating the probability of the mapping between them.

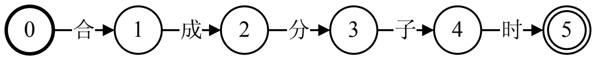
Given a group of WFSTs, each of which models a stage of a mapping cascade, the composition operation provides an efficient approach to combine them into a single one (Mohri et al., 2002; Mohri et al., 1996). In particular, for two WFSTs R and S, the composition $T = R^{o}S$ represents the composition of relations realized by R and S. The combination is performed strictly on R's output and S's input. It means for each path in T, mapping string r to string s, there must exist a path mapping r to some string t in R and a path mapping t to s in S. Decoding on the combined WFST enables to find the joint optimal results for multi-level weighted relations.

2.2 Model Unit Selection

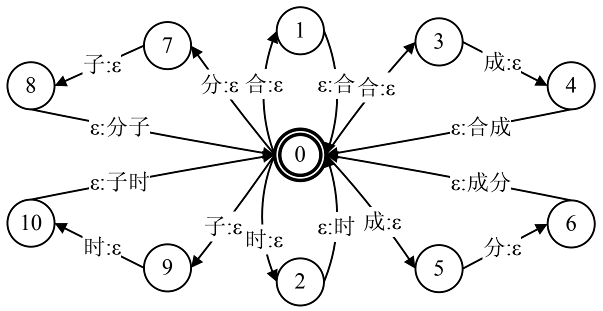
This study primarily takes the person and location names as well as the POS information into account. To deal with Chinese OOV words, different from the western language in which the phoneme, syllable or sub-word are used as the model units (Bazzi and Glass, 2000a; Bazzi and Glass, 2000b; Galescu, 2003), Chinese characters are taken as the basic units. In general, a person name of Han nationality consists of a surname and a given name usually with one or two characters. Surnames commonly come from a fixed set that has been historically used. According to a recent investigation on surnames involving 296 million people, 4100 surnames are found, and 129 most used surnames account for 87% (conducted by the Institute of Genetics and Developmental Biology, Chinese Academy of Sciences). In contrast, the characters used in given names can be selected freely, and in many situations, some commonly used words may also appear in names, such as ”胜利” (victory) and ”长江” (the Changjiang River). Therefore, both Chinese characters and words are considered as model units in this study, and a word re-segmentation process on recognition hypotheses is necessary, where an n-gram language model based on word classes is adopted.

2.3 Representation and Integration of Multi-level Knowledge

In this work, we ignore the word boundaries of n-best hypotheses and perform a word re-segmentation for names recognition. Given an input Chinese character, it is encoded by a finite state acceptor FSA_{input}. For example, the input ”合成分子时” (while synthesizing molecule) is represented as in Figure 1(a). Then a dictionary is represented by a



(a)



(b)

Figure 1: (a) is an example of the FSA representing a given input; (b) is the FST representing a toy dictionary.

transducer with empty weights, denoted as FST_{dict}. Figure 1(b) illustrates a toy dictionary listed in Table 1, in which a successful path encodes a mapping from a Chinese character sequence to some word in the dictionary. In practice, all Chinese characters should appear in the dictionary for further incorporating models of names. Then the combination of FSA_{input} and FST_{dict}, $FST_{seg} = FSA_{input} \circ FST_{dict}$, will result in a WFST embracing all the possible candidate segmentations. Afterwards an n-gram language model based on word classes is used to weight the candidate segmentations. As in Figure 2, a toy bigram with three words is depicted by $WFST_{n-gram}$, and the word classes are defined in Table 2. Here, both in the training and test stages, the strings of numbers or letters in sentences are extracted according to the rules, and then substituted with the class tags, ”NUM” and ”LETTER” respectively. At the same time, the words, such as ”三月” and ”A型”, are replaced with ”NUM月” and ”LETTER型” in the dictionary. In addition, name classes, including ”CNAME”, ”TNAME” and ”LOC”, will be set according to names recognition.

Chinese Words	English Words
合成	synthesize
成分	element
分子	molecule
子时	the period of the day from 11 p.m.to l a.m.
合	together
时	present

Table 1: The Toy dictionary

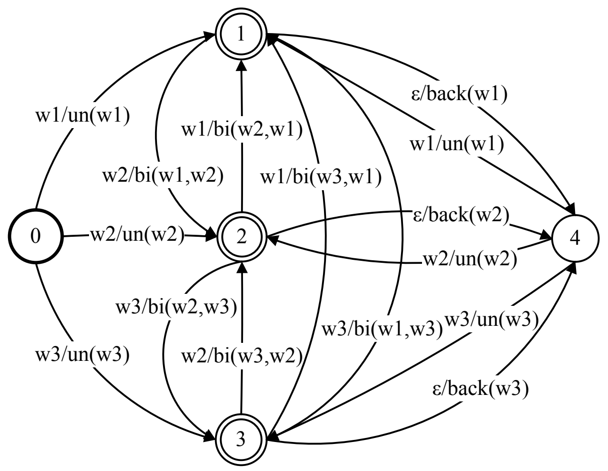


Figure 2: The WFST representing a toy bigram language model, in which $un(w1)$ denotes the unigram of $w1$; $bi(w1, w2)$ and $back(w1)$ respectively denotes the bigram of $w2$ and the backoff weight given the word history $w1$.

Classes	Description
w_i	Each word w_i listed in the dictionary
CNAME	Person names of Han nationality
TNAME	Translated person names
LOC	Location names
NUM	Number expressions
LETTER	Letter strings
NON	Other non Chinese character strings
BEGIN	Beginning of sentence
END	End of sentence

Table 2: The Definition of word classes

Hidden Markov Models (HMMs) are adopted both for names recognition and POS tagging. Here, each HMM is represented with two WFSTs. Taking the POS tagging as an example, the toy POS WFSTs with 3 different tags are illustrated in Figure 3. The emission probability of a word by a POS, ($P(word/pos)$), is represented as in Figure 3(a), and the bigram transition probabilities between POS tags are represented as in Figure 3(b), similar to the word n-gram. In terms of names recognition, the HMM states correspond to 30 role tags of names, some for model units of Chinese characters, such as surname, the first or second character of a given person name with two characters, the first or last character of a location name and so on, but others for model units of words, such as the word before or after a name, the words in a name and so on. When recognizing the person names, since there is a big difference between the translated names and the names of Han nationality, two types of person names are modeled separately, and substituted with two different class tags in the segmentation language model, as ”TNAME” and ”CNAME”. Some rules, which can be encoded into WFSTs, are responsible for the transformation from a role sequence to corresponding name class (for example, a role sequence might consist of the surname, the first character of the

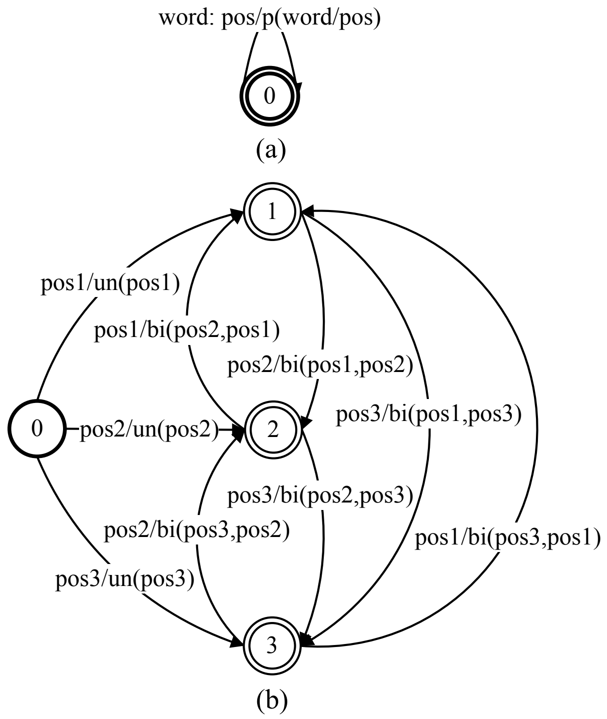


Figure 3: The toy POS WFSTs. (a) is the WFST representing the relationship between the word and the pos; (b) is the WFSA representing the bigram transition probabilities between POS tags

given name, and the second character of the given name, which will be transformed to "CNAME" in FST_{seg}). Hence, taking names recognition into account, a WFST, including all possible segmentations as well as recognized candidates of names, can be obtained as below, denoted as $WFST_{words}$:

$$FSA_{input} \circ FST_{dict} \circ WFST_{ne} \circ WFSA_{n-gram} \tag{1}$$

POS information is integrated as follows.

$$(\alpha * WFST_{words}) \circ WFST_{POS} \tag{2}$$

Consequently, the desired analyzer, a combined WFST that represents multi-level linguistic knowledge sources, has been obtained.

2.4 Incorporation in LVCSR

The presented analyzer models linguistic knowledge at different levels, which will be useful to find an optimal words sequence among a large number of speech recognition hypotheses. Thus in this research, the analyzer is incorporated after the first pass recognition, and the n-best hypotheses are reranked according to the total path scores adjusted with the analyzer scores as follows.

$$\hat{W} = \arg\max_{W} \left(\begin{array}{l} \log\left(P_{AM}\left(O|W\right)\right) \\ +\beta * \log\left(P_{LM}\left(W\right)\right) \\ +\gamma * \log\left(P_{Analyzer}\left(W\right)\right) \end{array} \right) \tag{3}$$

where $P_{AM}\left(O|W\right)$ and $P_{LM}\left(W\right)$ are the acoustic and language scores produced in first pass decoding, and $P_{Analyzer}\left(W\right)$ reflects the linguistic correctness of one hypothesis scored by the analyzer. Through the reranking paradigm, a new best sentence hypothesis is obtained.

3 Simulation

Under the unified framework, multi-level linguistic knowledge is represented by the analyzer as mentioned above. To guarantee the effectiveness of the introduced framework in integrating knowledge sources, the analyzer is evaluated in this section. Then the experiments using an LVCSR system in which the analyzer is embedded are performed.

3.1 Analyzer Evaluation

Considering the function of the analyzer, cascaded subtasks of word segmentation, names recognition and POS tagging can be processed jointly, while they are traditionally handled in a pipeline manner. Hence, a comparison between the analyzer and the pipeline system can be used to evaluate the effectiveness of the introduced framework for knowledge integration. As illustrated in Figure 4, two systems based on the presented analyzer and the pipeline manner are constructed respectively.

The evaluation data came from the People's Daily of China in 1998 from January to June (annotated by the Institute of Computational Linguistics of Peking University[1]), among which the January to May data was taken as the training set, and the June data was taken as the test set (consisted of 21,143 sentences and about 1.2 million words). The first two thousand sentences from the June data were extracted as the development set, used to fix the composition weight α in equation 2. A dictionary including about 113,000 words was extracted from the training data,

[1]http://icl.pku.edu.cn/icl_res/

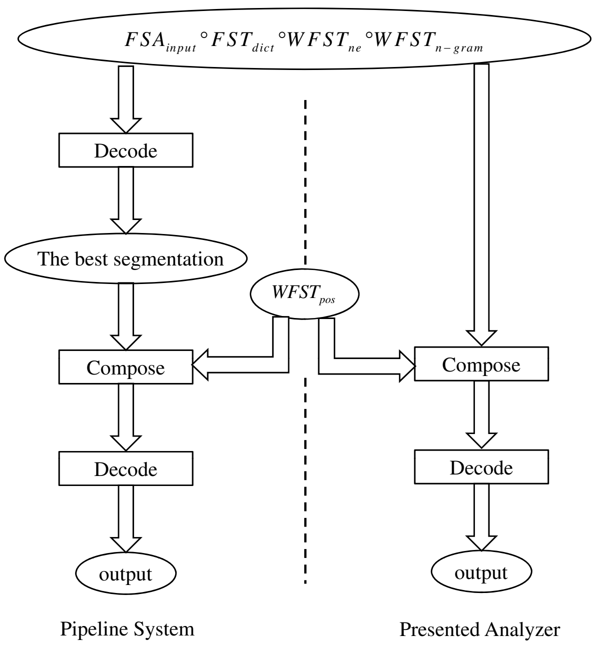


Figure 4: The pipeline system vs The analyzer

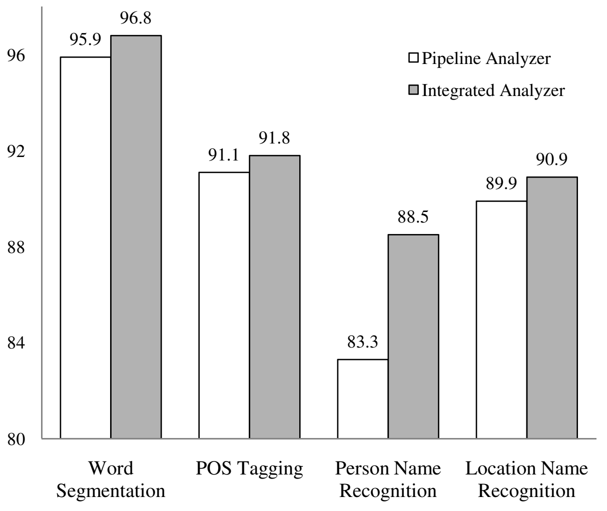


Figure 5: The Performance comparison between the pipeline system and the analyzer. The system performances are measured with the F1-score in the tasks of word segmentation, POS tagging, the person names recognition and the location names recognition.

in which a person or location name was accounted as a word in vocabulary, only when the number of its appearances was no less than three.

In Figure 5, the analyzer is compared with the pipeline system, where the analyzer outperforms the pipeline manner on all the subtasks in terms of $F1$-score metric. Furthermore to detect the differences, the statistical significance test using approximate randomization approach (Yeh, 2000) is done on the word segmentation results. Since there are more than 21,000 sentences in the test set, which is not appropriate for approximate randomization test, ten sets (500 sentences for each) are randomly selected from the test corpus. For each set, we run 1048576 shuffles twice and calculate the significance level p-value according to the shuffled results. It has been shown that all p-value are less than 0.001 on the ten sets. Accordingly the improvement is statistically significant. Actually, this significant improvement is reasonable, since the joint processing avoids error propagation and provides the opportunity of sharing information between different level knowledge sources. The superiority of this analyzer also shows that the integration of multi-level linguistic knowledge under the unified framework is effective, which may lead to improved LVCSR.

3.2 Experimental Setup for Mandarin Speech Recognition

In the baseline speech recognition system, the acoustic models consisted of context-dependent Initial-Final models, in which the left-to-right model topology was used to represent each unit. According to the phonetic structures, the number of states in each model was set to 2 or 3 for initials, and 4 or 5 for tonal finals. Each state was trained to have 32 Gaussian mixtures. The used 39-dimension feature vector comprised 12 MFCC coefficients, energy, and their first-order and second-order deltas. Since in this work we focused on modeling knowledge of language in Mandarin LVCSR, only clean male acoustic models were trained with a speech database that contained about 360 hours speech of over 750 male speakers. This training data was picked up from three continuous Mandarin speech corpora: the 863-I, 863-II and Intel corpora. The brief information about these three speech corpora was listed in Table 3. As in this work, the evaluation data was the 1997 HUB-4 Mandarin broadcast news evaluation data (HUB-4 test set), to better fit this task, the acoustic models were adapted by the approach of maximum a posterior (MAP) adaptation. The adaption data was drawn from the HUB4 training set, excluding the HUB-4 develop-

Corpus	Speakers	Amount of Speech (hours)
863-I (male)	83	56.67
863-II(male)	120	78.08
Intel (male)	556	227.30
total	759	362.05

Table 3: The information of the speech training data

ing set, where only the cleaned male speech data (data under condition f0 defined as (Doddington, 1996)) was used. The partition for the clean data was done with the acoustic segmentation software CMUseg_0.5[2] (Siegler et al., 1997), and finally 8.6 hours adaptation data was obtained.

The language model was a word-based trigram built on 60,000 words entries and trained with a corpus about 1.5 billion characters. The training set consisted of broadcast news data from the Xinhua News Agency released by LDC (Xinhua part of Chinese Gigaword), seven years data of People's Daily of China from 1995 to 2002 released by People's Daily Online[3], and some other data from news websites, such as yahoo, sina and so on.

In addition, the analyzer incorporated in speech recognition was trained with a larger corpus from People's Daily of China, including the data in 1998 from January to June and the data in 2000 from January to November (annotated by the Institute of Computational Linguistics of Peking University). The December data in 2000 was taken as the development set used to fix the composition weight α in equation 2.

3.3 Experimental Results

In our experiments, the clean male speech data from the Hub-4 test set was used, and 238 sentences were finally extracted for testing. The weight of the analyzer was empirically derived from the development set, including 649 clean male sentences from the devSet of HUB-4 Evaluation. The recognition results are shown in Table 4. The baseline system has a character error rate (CER) of 14.85%. When the analyzer is incorporated, a 9.9% relative reduction is achieved. Furthermore, we ran the statistical significance test to detect the performance improvement, in which the approximate randomization approach (Yeh, 2000) was modified to output the significance level, p-value, for the CER metric. The p-levels produced through two rounds of 1048576 shuffles are 0.0058 and 0.0057 respectively, both less than 0.01. Thus the performance improvement imposed by the utilization of the analyzer is statistically significant.

System	Err.	Sub.	Del.	Ins.
Baseline	14.85	13.02	0.76	1.07
Analyzer incorporation	13.38	11.78	1.00	0.60

Table 4: The Speech recognition results

4 Conclusion

Addressing the challenges of Mandarin large vocabulary continuous speech recognition task, within the unified framework of WFSTs, this study presents an analyzer integrating multi-level linguistic knowledge. Unlike other methods, model units, such as characters and words, can be chosen freely in this approach since multi-level knowledge sources are modeled independently. As a consequence, the final analyzer can be derived from the combination of better optimized models based on proper model units. Along with two level knowledge sources, i.e., the person and location names as well as the part-of-speech information, the analyzer is built and evaluated by a comparative simulation. Further evaluation is also conducted on an LVCSR system in which the analyzer is embedded. Experimental results consistently reveal that the approach is effective, and successfully improves the performance of speech recognition by a 9.9% relative reduction of character error rate on the HUB-4 test set. Also, the unified framework based approach provides a property of integrating additional linguistic knowledge flexibly, such as organization name and syntactic structure. Furthermore, the presented approach has a benefit of efficiency that additional efforts are not required for decoding as new knowledge comes, since all knowledge sources are finally encoded into a single WFST.

[2]Acoustic segmentation software downloaded from http://www.nist.gov/speech/tools/CMUseg_05targz.htm.

[3]http://www.people.com.cn

Acknowledgments

The work was supported in part by the National Natural Science Foundation of China (60435010; 60535030; 60605016), the National High Technology Research and Development Program of China (2006AA01Z196; 2006AA010103), the National Key Basic Research Program of China (2004CB318005), and the New-Century Training Program Foundation for the Talents by the Ministry of Education of China.

References

Steven Abney. 1996. Partial parsing via finite-state cascades. *Natural Language Engineering*, 2(4):337–344.

Issam Bazzi and James R. Glass. 2000a. Modeling out-of-vocabulary words for robust speech recognition. In *Proc. of 6th International Conference on Spoken Language Processing*, pages 401–404, Beijing, China, October.

Issam Bazzi and James Glass. 2000b. Heterogeneous lexical units for automatic speech recognition: preliminary investigations. In *Proc. of ICASSP*, pages 1257–1260, Istanbul, Turkey, June.

Issam Bazzi and James Glass. 2001. Learning units for domain-independent out-of-vocabulary word modelling. In *Proc. of EUROSPEECH*, pages 61–64, Aalborg, Denmark, September.

Eugene Charniak. 2001. Immediate-head parsing for language models. In *Proc. of ACL*, pages 116–123, Toulouse, France, July.

Ciprian Chelba, David Engle, Frederick Jelinek, Victor Jimenez, Sanjeev Khudanpur, Lidia Mangu, Harry Printz, Eric Ristad, Ronald Rosenfeld, Andreas Stolcke, and Dekai Wu. 1997. Structure and performance of a dependency language model. In *Proc. of EUROSPEECH*, pages 2775–2778, Rhodes, Greece.

Ciprian Chelba. 2000. *Exploiting Syntactic Structure for Natural Language Modeling*. Ph.D. thesis, Johns Hopkins University.

Isabel Trancoso Diamantino Caseiro. 2002. Using dynamic WFST composition for recognizing broadcast news. In *Proc. of ICSLP*, pages 1301–1304, Denver, Colorado, USA, September.

George Doddington. 1996. The 1996 hub-4 annotation specification for evaluation of speech recognition on broadcast news. In *ftp://jaguar.ncsl.nist.gov/csr96/h4/h4annot.ps*.

N. Friburger and D. Maurel. 2004. Finite-state transducer cascades to extract named entities in texts. *Theoretical Computer Science*, 313(1):93–104.

Lucian Galescu. 2003. Recognition of out-of-vocabulary words with sub-lexical language models. In *Proc. of EUROSPEECH*, pages 249–252, Geneva, Switzerland, September.

Thomas Hain, Lukas Burget, John Dines, Giulia Garau, Martin Karafiat, Mike Lincoln, Jithendra Vepa, and Vincent Wan. 2006. The AMI meeting transcription system: Progress and performance. In *Proc. of Rich Transcription 2006 Spring Meeting Recognition Evaluation*.

Peter A. Heeman. 1998. Pos tagging versus classes in language modeling. In *Proc. of the 6th Workshop on very large corpora*, pages 179–187, Montreal, Canada.

Takaaki Hori, Chiori Hori, Yasuhiro Minami, and Atsushi Nakamura. 2007. Efficient WFST-based one-pass decoding with on-the-fly hypothesis rescoring in extremely large vocabulary continuous speech recognition. *IEEE Transactions on audio, speech, and language processing*, 15(4):1352–1365.

Xinhui Hu, Hirofumi Yamamoto, Genichiro Kikui, and Yoshinori Sagisaka. 2006. Language modeling of chinese personal names based on character units for continuous chinese speech recognition. In *Proc. of INTERSPEECH*, pages 249–252, Pittsburgh, USA, September.

Mark Johnson. 2001. Joint and conditional estimation of tagging and parsing models. In *Proc. of ACL*, pages 322 – 329, Toulouse, France.

G. Maltese, P. Bravetti, H. Crépy, B. J. Grainger, M. Herzog, and F. Palou. 2001. Combining word- and class-based language models: A comparative study in several languages using automatic and manual word-clustering techniques. In *Proc. of EUROSPEECH*, pages 21–24, Aalborg, Denmark, September.

Ciro Martins, Antonio Texeira, and Joao Neto. 2006. Dynamic vocabulary adaptation for a daily and real-time broadcast news transcription system. In *Proc. of Spoken Language Technology Workshop*, pages 146–149, December.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 1996. Weighted automata in text and speech processing. In *ECAI-96 Workshop*.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2000. The design principles of a weighted finite-state transducer library. *Theoretical Computer Science*, 231(1):17–32.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2002. Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, 16(1):69–88.

Mehrya Mohri. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311.

Christan Raymond, Fre de ric Be chet, Renato D. Mori, and Ge raldine Damnati. 2006. On the use of finite state transducers for semantic interpretation. *Speech Communication*, 48(3-4):288–304.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

Christer Samuelsson and Wolfgang Reichl. 1999. A class-based language model for large-vocabulary speechrecognition extracted from part-of-speech statistics. In *Proc. of ICASSP*, pages 537–540, Phoenix, Arizona, USA, March.

George Saon, Geoffrey Zweig, Brain KingsBury, Lidia Mangu, and Upendra Canudhari. 2003. An architecture for rapid decoding of large vocabulary conversational speech. In *Proc. of Eurospeech*, pages 1977–1980, Geneva, Switzerland, September.

Matthew A. Siegler, Uday Jain, Bhiksha Raj, and Richard M. Stern. 1997. Automatic segmentation, classification and clustering of broadcast news audio. In *Proc. of DARPA Speech Recognition Workshop*, pages 97–99, Chantilly, Virginia, February.

Daniel Willett Takaaki Hori and Yasuhiro Minami. 2003. Language model adaptation using WFST-based speaking-style translation. In *Proc. of ICASSP*, pages I.228–I.231, Hong Kong, April.

Koichi Tanigaki, Hirofumi Yamamoto, and Yoshinori Sagisaka. 2000. A hierarchical language model incorporating class-dependent word models for oov words recognition. In *Proc. of 6th International Conference on Spoken Language Processing*, pages 123–126, Beijing, China, October.

Hajime Tsukada and Masaaki Nagata. 2004. Efficient decoding for statistical machine translation with a fully expanded WFST model. In *Proc. of EMNLP*, pages 427–433, Barcelona, Spain, July.

Wen Wang and Mary P. Harper. 2002. The superarv language model: investigating the effectiveness of tightly integrating multiple knowledge sources. In *Proc. of EMNLP*, pages 238–247, Philadelphia, USA, July.

Wen Wang and Dimitra Vergyri. 2006. The use of word n-grams and parts of speech for hierarchical cluster language modeling. In *Proc. of ICASSP*, pages 1057–1060, Toulouse, France, May.

Wen Wang, Andreas Stolcke, and Mary P. Harper. 2004. The use of a linguistically motivated language model in conversational speech recognition. In *Proc. of ICASSP*, pages 261–264, Montreal, Canada, May.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proc. of COLING*, pages 947–953, Saarbrücken, August.

N-gram Weighting: Reducing Training Data Mismatch in Cross-Domain Language Model Estimation

Bo-June (Paul) Hsu, James Glass
MIT Computer Science and Artificial Intelligence Laboratory
32 Vassar Street, Cambridge, MA, 02139 USA
{bohsu,glass}@csail.mit.edu

Abstract

In domains with insufficient matched training data, language models are often constructed by interpolating component models trained from partially matched corpora. Since the n-grams from such corpora may not be of equal relevance to the target domain, we propose an n-gram weighting technique to adjust the component n-gram probabilities based on features derived from readily available segmentation and metadata information for each corpus. Using a log-linear combination of such features, the resulting model achieves up to a 1.2% absolute word error rate reduction over a linearly interpolated baseline language model on a lecture transcription task.

1 Introduction

Many application domains in machine learning suffer from a dearth of matched training data. However, partially matched data sets are often available in abundance. Past attempts to utilize the mismatched data for training often result in models that exhibit biases not observed in the target domain. In this work, we will investigate the use of the often readily available data segmentation and metadata attributes associated with each corpus to reduce the effect of such bias. We will examine this approach in the context of language modeling for lecture transcription.

Compared with other types of audio data, lecture speech often exhibits a high degree of spontaneity and focuses on narrow topics with special terminologies (Glass et al., 2004). While we may have existing transcripts from general lectures or written text on the precise topic, training data that matches both the style and topic of the target lecture is often scarce. Thus, past research has investigated various adaptation and interpolation techniques to make use of partially matched corpora (Bellegarda, 2004).

Training corpora are often segmented into documents with associated metadata, such as title, date, and speaker. For lectures, if the data contains even a few lectures on linear algebra, conventional language modeling methods that lump the documents together will tend to assign disproportionately high probability to frequent terms like *vector* and *matrix*. Can we utilize the segmentation and metadata information to reduce the biases resulting from training data mismatch?

In this work, we present such a technique where we weight each n-gram count in a standard n-gram language model (LM) estimation procedure by a relevance factor computed via a log-linear combination of n-gram features. Utilizing features that correlate with the specificity of n-grams to subsets of the training documents, we effectively de-emphasize out-of-domain n-grams. By interpolating models, such as general lectures and course textbook, that match the target domain in complementary ways, and optimizing the weighting and interpolation parameters jointly, we allow each n-gram probability to be modeled by the most relevant interpolation component. Using a combination of features derived from multiple partitions of the training documents, the resulting weighted n-gram model achieves up to a 1.2% absolute word error rate (WER) reduction over a linearly interpolated baseline on a lecture transcription task.

2 Related Work

To reduce topic mismatch in LM estimation, we (2006) have previously assigned topic labels to each word by applying HMM-LDA (Griffiths et al., 2005) to the training documents. Using an ad hoc method to reduce the effective counts of n-grams ending on topic words, we achieved better perplexity and WER than standard trigram LMs. Intuitively, de-emphasizing such n-grams will lower the transition probability to out-of-domain topic words from the training data. In this work, we further explore this intuition with a principled feature-based model, integrated with LM smoothing and estimation to allow simultaneous optimization of all model parameters.

As Gao and Lee (2000) observed, even purported matched training data may exhibit topic, style, or temporal biases not present in the test set. To address the mismatch, they partition the training documents by their metadata attributes and compute a measure of the likelihood that an n-gram will appear in a new partitioned segment. By pruning n-grams with generality probability below a given threshold, the resulting model achieves lower perplexity than a count-cutoff model of equal size. Complementary to our work, this technique also utilizes segmentation and metadata information. However, our model enables the simultaneous use of all metadata attributes by combining features derived from different partitions of the training documents.

3 N-gram Weighting

Given a limited amount of training data, an n-gram appearing frequently in a single document may be assigned a disproportionately high probability. For example, an LM trained from lecture transcripts tends to assign excessive probability to words from observed lecture topics due to insufficient coverage of the underlying document topics. On the other hand, excessive probabilities may also be assigned to n-grams appearing consistently across documents with mismatched style, such as the course textbook in the written style. Traditional n-gram smoothing techniques do not address such issues of insufficient topic coverage and style mismatch.

One approach to addressing the above issues is to weight the counts of the n-grams according to the concentration of their document distributions. Assigning higher weights to n-grams with evenly spread distributions captures the style of a data set, as reflected across all documents. On the other hand, emphasizing the n-grams concentrated within a few documents focuses the model on the topics of the individual documents.

In theory, n-gram weighting can be applied to any smoothing algorithm based on counts. However, because many of these algorithms assume integer counts, we will apply the weighting factors to the smoothed counts, instead. For modified Kneser-Ney smoothing (Chen and Goodman, 1998), applying n-gram weighting yields:

$$p(w|h) = \frac{\beta(hw)\tilde{c}'(hw)}{\sum_w \beta(hw)\tilde{c}(hw)} + \alpha(h)p(w|h')$$

where $p(w|h)$ is the probability of word w given history h, $\tilde{c}$ is the adjusted Kneser-Ney count, $\tilde{c}'$ is the discounted count, β is the n-gram weighting factor, α is the normalizing backoff weight, and h' is the backoff history.

Although the weighting factor β can in general be any function of the n-gram, in this work, we will consider a log-linear combination of n-gram features, or $\beta(hw) = \exp(\mathbf{\Phi}(hw) \cdot \boldsymbol{\theta})$, where $\mathbf{\Phi}(hw)$ is the feature vector for n-gram hw and $\boldsymbol{\theta}$ specifies the parameter vector to be learned. To better fit the data, we allow independent parameter vectors $\boldsymbol{\theta}_o$ for each n-gram order o. Note that with $\beta(hw) = 1$, the model degenerates to the original modified Kneser-Ney formulation. Furthermore, β only specifies the relative weighting among n-grams with a common history h. Thus, scaling $\beta(hw)$ by an arbitrary function $g(h)$ has no effect on the model.

In isolation, n-gram weighting shifts probability mass from out-of-domain n-grams via backoff to the uniform distribution to improve the generality of the resulting model. However, in combination with LM interpolation, it can also distribute probabilities to LM components that better model specific n-grams. For example, n-gram weighting can de-emphasize off-topic and off-style n-grams from general lectures and course textbook, respectively. Tuning the weighting and interpolation parameters jointly further allows the estimation of the n-gram probabilities to utilize the best matching LM components.

3.1 Features

To address the issue of sparsity in the document topic distribution, we can apply n-gram weighting with features that measure the concentration of the n-gram distribution across documents. Similar features can also be computed from documents partitioned by their categorical metadata attributes, such as *course* and *speaker* for lecture transcripts. Whereas the features derived from the corpus documents should correlate with the topic specificity of the n-grams, the same features computed from the speaker partitions might correspond to the speaker specificity. By combining features from multiple partitions of the training data to compute the weighting factors, n-gram weighting allows the resulting model to better generalize across categories.

To guide the presentation of the n-gram features below, we will consider the following example partition of the training corpus. Words tagged by HMM-LDA as topic words appear in bold.

A B A	A **C** **C** A	**B** A B
B A **A** C	C B A	A B A
A C B	A A **C**	A **B** **B** A

One way to estimate the specificity of an n-gram across partitions is to measure the n-gram frequency f, or the fraction of partitions containing an n-gram. For instance, $f(\text{A}) = 3/3$, $f(\text{C}) = 2/3$. However, as the size of each partition increases, this ratio increases to 1, since most n-grams have a non-zero probability of appearing in each partition. Thus, an alternative is to compute the normalized entropy of the n-gram distribution across the S partitions, or $h = \frac{-1}{\log S} \sum_{s=1}^{S} p(s) \log p(s)$, where $p(s)$ is the fraction of an n-gram appearing in partition s. For example, the normalized entropy of the unigram C is $h(\text{C}) = \frac{-1}{\log 3}[\frac{2}{6} \log \frac{2}{6} + \frac{4}{6} \log \frac{4}{6} + 0] = .58$. N-grams clustered in fewer partitions have lower entropy than ones that are more evenly spread out.

Following (Hsu and Glass, 2006), we also consider features derived from the HMM-LDA word topic labels.[1] Specifically, we compute the empirical probability t that the target word of the n-gram is labeled as a topic word. In the example corpus, $t(\text{C}) = 3/6$, $t(\text{A C}) = 2/4$.

[1]HMM-LDA is performed using 20 states and 50 topics with a 3rd-order HMM. Hyperparameters are sampled with a log-normal Metropolis proposal. The model with the highest likelihood from among 10,000 iterations of Gibbs sampling is used.

Feature	of the	i think	k means	the sun	this is a	a lot of	big o of	e m f
Random	0.03	0.32	0.33	0.19	0.53	0.24	0.37	0.80
$\log(c)$	9.29	8.09	3.47	5.86	6.82	7.16	3.09	4.92
f^{doc}	1.00	0.93	0.00	0.18	0.92	0.76	0.00	0.04
f^{course}	1.00	1.00	0.06	0.56	0.94	0.94	0.06	0.06
f^{speaker}	0.83	0.70	0.00	0.06	0.41	0.55	0.01	0.00
h^{doc}	0.96	0.84	0.00	0.56	0.93	0.85	0.00	0.34
h^{course}	0.75	0.61	0.00	0.55	0.78	0.65	0.00	0.00
h^{speaker}	0.76	0.81	0.00	0.09	0.65	0.80	0.12	0.00
t^{doc}	0.00	0.00	0.91	1.00	0.01	0.00	0.00	0.04
t^{course}	0.00	0.00	0.88	0.28	0.01	0.00	0.00	1.00
t^{speaker}	0.00	0.00	0.94	0.92	0.01	0.00	0.09	0.99

Table 1: A list of n-gram weighting features. f: n-gram frequency, h: normalized entropy, t: topic probability.

All of the above features can be computed for any partitioning of the training data. To better illustrate the differences, we compute the features on a set of lecture transcripts (see Section 4.1) partitioned by lecture (doc), course, and speaker. Furthermore, we include the log of the n-gram counts c and random values between 0 and 1 as baseline features. Table 1 lists all the features examined in this work and their values on a select subset of n-grams.

3.2 Training

To tune the n-gram weighting parameters $\boldsymbol{\theta}$, we apply Powell's method (Press et al., 2007) to numerically minimize the development set perplexity (Hsu and Glass, 2008). Although there is no guarantee against converging to a local minimum when jointly tuning both the n-gram weighting and interpolation parameters, we have found that initializing the parameters to zero generally yields good performance.

4 Experiments

4.1 Setup

In this work, we evaluate the perplexity and WER of various trigram LMs trained with n-gram weighting on a lecture transcription task (Glass et al., 2007). The target data consists of 20 lectures from an introductory computer science course, from which we withhold the first 10 lectures for the development

Dataset	# Words	# Sents	# Docs
Textbook	131,280	6,762	271
Lectures	1,994,225	128,895	230
Switchboard	3,162,544	262,744	4,876
CS Dev	93,353	4,126	10
CS Test	87,527	3,611	10

Table 2: Summary of evaluation corpora.

	Perplexity		WER	
Model	Dev	Test	Dev	Test
FixKN(1)	174.7	196.7	34.9%	36.8%
+ W(h^{doc})	172.9	194.8	34.7%	36.7%
FixKN(3)	168.6	189.3	34.9%	36.9%
+ W(h^{doc})	166.8	187.8	34.6%	36.6%
FixKN(10)	167.5	187.6	35.0%	37.2%
+ W(h^{doc})	165.3	185.8	34.7%	36.8%
KN(1)	169.7	190.4	35.0%	37.0%
+ W(h^{doc})	167.3	188.2	34.8%	36.7%
KN(3)	163.4	183.1	35.0%	37.1%
+ W(h^{doc})	161.1	181.2	34.7%	36.8%
KN(10)	162.3	181.8	35.1%	37.1%
+ W(h^{doc})	160.1	180.0	34.8%	36.8%

Table 3: Performance of n-gram weighting with a variety of Kneser-Ney settings. FixKN(d): Kneser-Ney with d fixed discount parameters. KN(d): FixKN(d) with tuned values. W(*feat*): n-gram weighting with *feat* feature.

set (CS Dev) and use the last 10 for the test set (CS Test). For training, we will consider the course textbook with topic-specific vocabulary (Textbook), numerous high-fidelity transcripts from a variety of general seminars and lectures (Lectures), and the out-of-domain LDC Switchboard corpus of spontaneous conversational speech (Switchboard) (Godfrey and Holliman, 1993). Table 2 summarizes all the evaluation data.

To compute the word error rate, we use a speaker-independent speech recognizer (Glass, 2003) with a large-margin discriminative acoustic model (Chang, 2008). The lectures are pre-segmented into utterances via forced alignment against the reference transcripts (Hazen, 2006). Since all the models considered in this work can be encoded as n-gram backoff models, they are applied directly during the first recognition pass instead of through a subsequent n-best rescoring step.

Model	Perplexity	WER
Lectures	189.3	36.9%
+ W(h^{doc})	187.8 (-0.8%)	36.6%
Textbook	326.1	43.1%
+ W(h^{doc})	317.5 (-2.6%)	43.1%
LI(Lectures + Textbook)	141.6	33.7%
+ W(h^{doc})	136.6 (-3.5%)	32.7%

Table 4: N-gram weighting with linear interpolation.

4.2 Smoothing

In Table 3, we compare the performance of n-gram weighting with the h^{doc} document entropy feature for various modified Kneser-Ney smoothing configurations (Chen and Goodman, 1998) on the Lectures dataset. Specifically, we considered varying the number of discount parameters per n-gram order from 1 to 10. The original and modified Kneser-Ney smoothing algorithms correspond to a setting of 1 and 3, respectively. Furthermore, we explored using both fixed parameter values estimated from n-gram count statistics and tuned values that minimize the development set perplexity.

In this task, while the test set perplexity tracks the development set perplexity well, the WER correlates surprisingly poorly with the perplexity on both the development and test sets. Nevertheless, n-gram weighting consistently reduces the absolute test set WER by a statistically significant average of 0.3%, according to the Matched Pairs Sentence Segment Word Error test (Pallet et al., 1990). Given that we obtained the lowest development set WER with the fixed 3-parameter modified Kneser-Ney smoothing, all subsequent experiments are conducted using this smoothing configuration.

4.3 Linear Interpolation

Applied to the Lectures dataset in isolation, n-gram weighting with the h^{doc} feature reduces the test set WER by 0.3% by de-emphasizing the probability contributions from off-topic n-grams and shifting their weights to the backoff distributions. Ideally though, such weights should be distributed to on-topic n-grams, perhaps from other LM components.

In Table 4, we present the performance of applying n-gram weighting to the Lectures and Textbook models individually versus in combination via linear interpolation (LI), where we optimize the n-gram

Model	Perplexity	WER
LI(Lectures + Textbook)	141.6	33.7%
+ W(*Random*)	141.5 (-0.0%)	33.7%
+ W($\log(c)$)	137.5 (-2.9%)	32.8%
+ W(f^{doc})	136.3 (-3.7%)	32.8%
+ W(f^{course})	136.5 (-3.6%)	32.7%
+ W(f^{speaker})	138.1 (-2.5%)	33.0%
+ W(h^{doc})	136.6 (-3.5%)	**32.7%**
+ W(h^{course}) ★	136.1 (-3.9%)	<u>32.7%</u>
+ W(h^{speaker})	138.6 (-2.1%)	33.1%
+ W(t^{doc})	**<u>134.8</u>** (-4.8%)	33.2%
+ W(t^{course})	136.4 (-3.6%)	33.1%
+ W(t^{speaker})	136.4 (-3.7%)	33.2%

Table 5: N-gram weighting with various features.

weighting and interpolation parameters jointly. The interpolated model with n-gram weighting achieves perplexity improvements roughly additive of the reductions obtained with the individual models. However, the 1.0% WER drop for the interpolated model significantly exceeds the sum of the individual reductions. Thus, as we will examine in more detail in Section 5.1, n-gram weighting allows probabilities to be shifted from less relevant n-grams in one component to more specific n-grams in another.

4.4 Features

With n-gram weighting, we can model the weighting function $\beta(hw)$ as a log-linear combination of any n-gram features. In Table 5, we show the effect various features have on the performance of linearly interpolating Lectures and Textbook. As the documents from the Lectures dataset is annotated with course and speaker metadata attributes, we include the n-gram frequency f, entropy h, and topic probability t features computed from the lectures grouped by the 16 unique courses and 299 unique speakers.[2]

In terms of perplexity, the use of the *Random* feature has negligible impact on the test set performance, as expected. On the other hand, the $\log(c)$ count feature reduces the perplexity by nearly 3%, as it correlates with the generality of the n-grams. By using features that leverage the information from document segmentation and associated metadata, we are generally able to achieve further perplexity reductions. Overall, the frequency and entropy features perform roughly equally. However, by considering information from the more sophisticated HMM-LDA topic model, the topic probability feature t^{doc} achieves significantly lower perplexity than any other feature in isolation.

In terms of WER, the *Random* feature again shows no effect on the baseline WER of 33.7%. However, to our surprise, the use of the simple $\log(c)$ feature achieves nearly the same WER improvement as the best segmentation-based feature, whereas the more sophisticated features computed from HMM-LDA labels only obtain half of the reduction even though they have the best perplexities.

When comparing the performance of different n-gram weighting features on this data set, the perplexity correlates poorly with the WER, on both the development and test sets. Fortunately, the features that yield the lowest perplexity and WER on the development set also yield one of the lowest perplexities and WERs, respectively, on the test set. Thus, during feature selection for speech recognition applications, we should consider the development set WER. Specifically, since the differences in WER are often statistically insignificant, we will select the feature that minimizes the sum of the development set WER and log perplexity, or cross-entropy.[3]

In Tables 5 and 6, we have underlined the perplexities and WERs of the features with the lowest corresponding development set values (not shown) and bolded the lowest test set values. The features that achieve the lowest combined cross-entropy and WER on the development set are starred.

4.5 Feature Combination

Unlike most previous work, n-gram weighting enables a systematic integration of features computed from multiple document partitions. In Table 6, we compare the performance of various feature combinations. We experiment with incrementally adding features that yield the lowest combined development set cross-entropy and WER. Overall, this metric appears to better predict the test set WER than either the development set perplexity or WER alone.

[2]Features that are not applicable to a particular corpus (e.g. h^{course} for Textbook) are removed from the n-gram weighting computation for that component. Thus, models with course and speaker features have fewer tunable parameters than the others.

[3]The choice of cross-entropy instead of perplexity is partially motivated by the linear correlation reported by (Chen and Goodman, 1998) between cross-entropy and WER.

Features	Perplexity	WER
h^{course}	136.1	32.7%
+ $\log(c)$	135.4 (-0.5%)	32.6%
+ f^{doc}	135.1 (-0.7%)	32.6%
+ h^{doc}	135.6 (-0.5%)	32.6%
+ t^{doc} ★	**133.2** (-2.1%)	**32.6%**
+ f^{course}	136.0 (-0.1%)	32.6%
+ t^{course}	134.8 (-1.0%)	32.9%
+ f^{speaker}	136.0 (-0.1%)	32.6%
+ h^{speaker}	136.1 (-0.0%)	32.8%
+ t^{speaker}	134.7 (-1.0%)	32.7%
h^{course} + t^{doc}	133.2	32.6%
+ $\log(c)$	132.8 (-0.3%)	32.5%
+ f^{doc} ★	**132.8** (-0.4%)	**32.5%**
+ h^{doc}	133.0 (-0.2%)	32.5%
+ f^{course}	133.1 (-0.1%)	32.5%
+ t^{course}	133.0 (-0.1%)	32.6%
+ f^{speaker}	133.1 (-0.1%)	32.5%
+ h^{speaker}	133.2 (-0.0%)	32.6%
+ t^{speaker}	133.1 (-0.1%)	32.7%

Table 6: N-gram weighting with feature combinations.

Model	Perplexity	WER
Linear(L + T)	141.6	33.7%
+ W(h^{course})	136.1 (-3.9%)	32.7%
+ W(t^{doc})	133.2 (-5.9%)	32.6%
+ W(f^{doc})	132.8 (-6.2%)	32.5%
CM(L + T)	137.9	33.0%
+ W(h^{course})	135.5 (-1.8%)	32.4%
+ W(t^{doc})	133.4 (-3.3%)	32.4%
+ W(f^{doc})	133.2 (-3.5%)	32.4%
GLI$_{\log(1+\tilde{c})}$(L + T)	135.9	33.0%
+ W(h^{course})	133.0 (-2.2%)	32.4%
+ W(t^{doc})	130.6 (-3.9%)	32.4%
+ W(f^{doc})	130.5 (-4.2%)	32.4%

Table 7: Effect of interpolation technique. L: Lectures, T: Textbook.

Feature	Parameter Values
h^{doc}	$\boldsymbol{\theta}^{\mathsf{L}} = [3.42, 1.46, 0.12]$
	$\boldsymbol{\theta}^{\mathsf{T}} = [-0.45, -0.35, -0.73]$
	$[\lambda^{\mathsf{L}}, \lambda^{\mathsf{T}}] = [0.67, 0.33]$
t^{doc}	$\boldsymbol{\theta}^{\mathsf{L}} = [-2.33, -1.63, -1.19]$
	$\boldsymbol{\theta}^{\mathsf{T}} = [1.05, 0.46, 0.12]$
	$[\lambda^{\mathsf{L}}, \lambda^{\mathsf{T}}] = [0.68, 0.32]$

Table 8: N-gram weighting parameter values. $\boldsymbol{\theta}^{\mathsf{L}}$, $\boldsymbol{\theta}^{\mathsf{T}}$: parameters for each order of the Lectures and Textbook trigram models, λ^{L},λ^{T}: linear interpolation weights.

Using the combined feature selection technique, we notice that the greedily selected features tend to differ in the choice of document segmentation and feature type, suggesting that n-gram weighting can effectively integrate the information provided by the document metadata. By combining features, we are able to further reduce the test set WER by a statistically significant ($p < 0.001$) 0.2% over the best single feature model.

4.6 Advanced Interpolation

While n-gram weighting with all three features is able to reduce the test set WER by 1.2% over the linear interpolation baseline, linear interpolation is not a particularly effective interpolation technique. In Table 7, we compare the effectiveness of n-gram weighting in combination with better interpolation techniques, such as count merging (CM) (Bacchiani et al., 2006) and generalized linear interpolation (GLI) (Hsu, 2007). As expected, the use of more sophisticated interpolation techniques decreases the perplexity and WER reductions achieved by n-gram weighting by roughly half for a variety of feature combinations. However, all improvements remain statistically significant.

Although the WER reductions from better interpolation techniques are initially statistically significant, as we add features to n-gram weighting, the differences among the interpolation methods shrink significantly. With all three features combined, the test set WER difference between linear interpolation and generalized linear interpolation loses its statistical significance. In fact, we can obtain statistically the same WER of 32.4% using the simpler model of count merging and n-gram weighting with h^{course}.

5 Analysis

5.1 Weighting Parameters

To obtain further insight into how n-gram weighting improves the resulting n-gram model, we present in Table 8 the optimized parameter values for the linear interpolation model between Lectures and Textbook using n-gram weighting with h^{doc} and t^{doc} features. Using $\beta(hw) = \exp(\boldsymbol{\Phi}(hw) \cdot \boldsymbol{\theta})$ to model the n-gram weights, a positive value of θ_i corresponds to

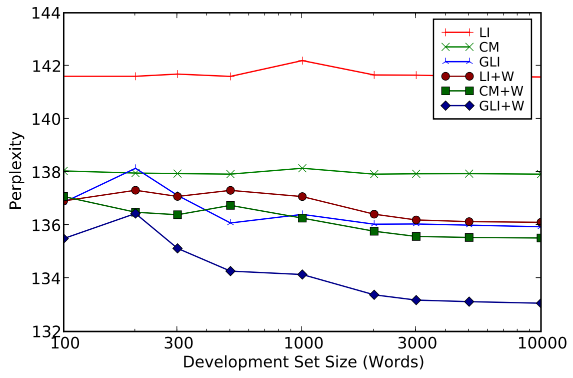


Figure 1: Test set perplexity vs. development set size.

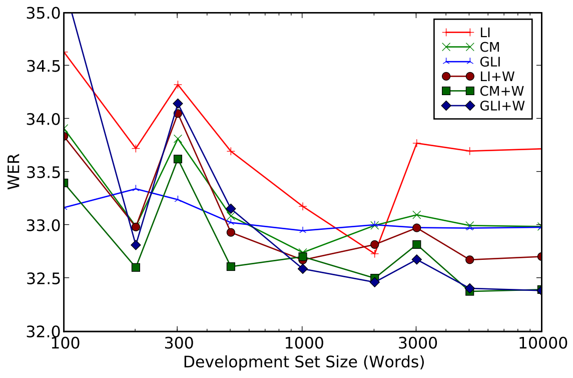


Figure 2: Test set WER vs. development set size.

increasing the weights of the i^{th} order n-grams with positive feature values.

For the h^{doc} normalized entropy feature, values close to 1 correspond to n-grams that are evenly distributed across the documents. When interpolating Lectures and Textbook, we obtain consistently positive values for the Lectures component, indicating a de-emphasis on document-specific terms that are unlikely to be found in the target computer science domain. On the other hand, the values corresponding to the Textbook component are consistently negative, suggesting a reduced weight for mismatched style terms that appear uniformly across textbook sections.

For t^{doc}, values close to 1 correspond to n-grams ending frequently on topic words with uneven distribution across documents. Thus, as expected, the signs of the optimized parameter values are flipped. By de-emphasizing topic n-grams from off-topic components and style n-grams from off-style components, n-gram weighting effectively improves the performance of the resulting language model.

5.2 Development Set Size

So far, we have assumed the availability of a large development set for parameter tuning. To obtain a sense of how n-gram weighting performs with smaller development sets, we randomly select utterances from the full development set and plot the test set perplexity in Figure 1 as a function of the development set size for various modeling techniques.

As expected, GLI outperforms both LI and CM. However, whereas LI and CM essentially converge in test set perplexity with only 100 words of development data, it takes about 500 words before GLI converges due to the increased number of parameters. By adding n-gram weighting with the h^{course} feature, we see a significant drop in perplexity for all models at all development set sizes. However, the performance does not fully converge until 3,000 words of development set data.

As shown in Figure 2, the test set WER behaves more erratically, as the parameters are tuned to minimize the development set perplexity. Overall, n-gram weighting decreases the WER significantly, except when applied to GLI with less than 1000 words of development data when the perplexity of GLI has not itself converged. In that range, CM with n-gram weighting performs the best. However, with more development data, GLI with n-gram weighting generally performs slightly better. From these results, we conclude that although n-gram weighting increases the number of tuning parameters, they are effective in improving the test set performance even with only 100 words of development set data.

5.3 Training Set Size

To characterize the effectiveness of n-gram weighting as a function of the training set size, we evaluate the performance of various interpolated models with increasing subsets of the Lectures corpus and the full Textbook corpus. Overall, every doubling of the number of training set documents decreases both the test set perplexity and WER by approximately 7 points and 0.8%, respectively. To better compare results, we plot the performance difference between various models and linear interpolation in Figures 3 and 4.

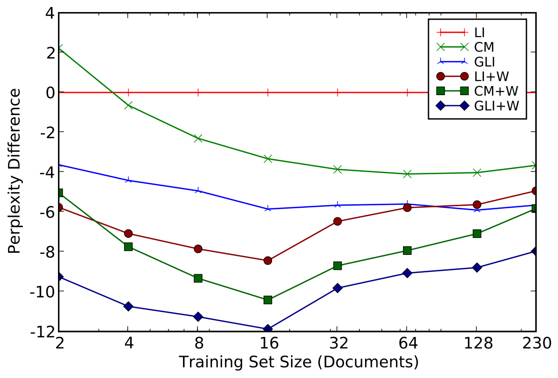


Figure 3: Test set perplexity vs. training set size.

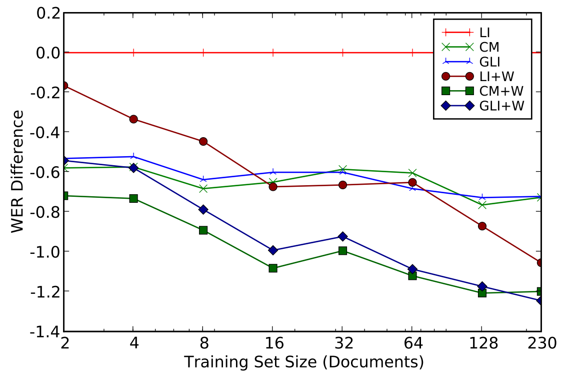


Figure 4: Test set WER vs. training set size.

Interestingly, the peak gain obtained from n-gram weighting with the h^{doc} feature appears at around 16 documents for all interpolation techniques. We suspect that as the number of documents initially increases, the estimation of the h^{doc} features improves, resulting in larger perplexity reduction from n-gram weighting. However, as the diversity of the training set documents increases beyond a certain threshold, we experience less document-level sparsity. Thus, we see decreasing gain from n-gram weighting beyond 16 documents.

For all interpolation techniques, even though the perplexity improvements from n-gram weighting decrease with more documents, the WER reductions actually increase. N-gram weighting showed statistically significant reductions for all configurations except generalized linear interpolation with less than 8 documents. Although count merging with n-gram weighting has the lowest WER for most training set sizes, GLI ultimately achieves the best test set WER with the full training set.

5.4 Training Corpora

In Table 9, we compare the performance of n-gram weighting with different combination of training corpora and interpolation techniques to determine its effectiveness across different training conditions. With the exception of interpolating Lectures and Switchboard using count merging, all other model combinations yield statistically significant improvements with n-gram weighting using h^{course}, t^{doc}, and f^{doc} features.

The results suggest that n-gram weighting with these features is most effective when interpolating

Model	L + T	L + S	T + S	L + T + S
LI	33.7%	36.7%	36.4%	33.6%
LI + W	32.5%	36.4%	35.7%	32.5%
CM	33.0%	36.6%	35.5%	32.9%
CM + W	32.4%	36.5%	35.4%	32.3%
GLI	33.0%	36.6%	35.7%	32.8%
GLI + W	32.4%	36.4%	35.3%	32.2%

Table 9: Test set WER with various training corpus combinations. L: Lectures, T: Textbook, S: Switchboard, W: n-gram weighting.

corpora that differ in how they match the target domain. Whereas the Textbook corpus is the only corpus with matching topic, both Lectures and Switchboard have a similar matching spoken conversational style. Thus, we see the least benefit from n-gram weighting when interpolating Lectures and Switchboard. By combining Lectures, Textbook, and Switchboard using generalized linear interpolation with n-gram weighting using h^{course}, t^{doc}, and f^{doc} features, we achieve our best test set WER of 32.2% on the lecture transcription task, a full 1.5% over the initial linear interpolation baseline.

6 Conclusion & Future Work

In this work, we presented the n-gram weighting technique for adjusting the probabilities of n-grams according to a set of features. By utilizing features derived from the document segmentation and associated metadata inherent in many training corpora, we achieved up to a 1.2% and 0.6% WER reduction over the linear interpolation and count merging baselines, respectively, using n-gram weighting on a lecture transcription task.

We examined the performance of various n-gram weighting features and generally found entropy-based features to offer the best predictive performance. Although the topic probability features derived from HMM-LDA labels yield additional improvements when applied in combination with the normalized entropy features, the computational cost of performing HMM-LDA may not justify the marginal benefit in all scenarios.

In situations where the document boundaries are unavailable or when finer segmentation is desired, automatic techniques for document segmentation may be applied (Malioutov and Barzilay, 2006). Synthetic metadata information may also be obtained via clustering techniques (Steinbach et al., 2000). Although we have primarily focused on n-gram weighting features derived from segmentation information, it is also possible to consider other features that correlate with n-gram relevance.

N-gram weighting and other approaches to cross-domain language modeling require a matched development set for model parameter tuning. Thus, for future work, we plan to investigate the use of the initial recognition hypotheses as the development set, as well as manually transcribing a subset of the test set utterances.

As speech and natural language applications shift towards novel domains with limited matched training data, better techniques are needed to maximally utilize the often abundant partially matched data. In this work, we examined the effectiveness of the n-gram weighting technique for estimating language models in these situations. With similar investments in acoustic modeling and other areas of natural language processing, we look forward to an ever increasing diversity of practical speech and natural language applications.

Availability An implementation of the n-gram weighting algorithm is available in the MIT Language Modeling (MITLM) toolkit (Hsu and Glass, 2008): http://www.sls.csail.mit.edu/mitlm/.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive feedback. This research is supported in part by the T-Party Project, a joint research program between MIT and Quanta Computer Inc.

References

Michiel Bacchiani, Michael Riley, Brian Roark, and Richard Sproat. 2006. MAP adaptation of stochastic grammars. *Computer Speech & Language*, 20(1):41–68.

Jerome R. Bellegarda. 2004. Statistical language model adaptation: Review and perspectives. *Speech Communication*, 42(1):93–108.

Hung-An Chang. 2008. Large margin Gaussian mixture modeling for automatic speech recognition. Massachusetts Institute of Technology. Masters Thesis.

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. In *Technical Report TR-10-98*. Computer Science Group, Harvard University.

Jianfeng Gao and Kai-Fu Lee. 2000. Distribution-based pruning of backoff language models. In *Proc. Association of Computational Linguistics*, pages 579–588, Hong Kong, China.

James Glass, Timothy J. Hazen, Lee Hetherington, and Chao Wang. 2004. Analysis and processing of lecture audio data: Preliminary investigations. In *Proc. HLT-NAACL Workshop on Interdisciplinary Approaches to Speech Indexing and Retrieval*, pages 9–12, Boston, MA, USA.

James Glass, Timothy J. Hazen, Scott Cyphers, Igor Malioutov, David Huynh, and Regina Barzilay. 2007. Recent progress in the MIT spoken lecture processing project. In *Proc. Interspeech*, pages 2553–2556, Antwerp, Belgium.

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

John J. Godfrey and Ed Holliman. 1993. Switchboard-1 transcripts. Linguistic Data Consortium, Philadelphia, PA, USA.

Thomas L. Griffiths, Mark Steyvers, David M. Blei, and Joshua B. Tenenbaum. 2005. Integrating topics and syntax. In *Advances in Neural Information Processing Systems 17*, pages 537–544. MIT Press, Cambridge, MA, USA.

T.J. Hazen. 2006. Automatic alignment and error correction of human generated transcripts for long speech recordings. In *Proc. Interspeech*, Pittsburgh, PA, USA.

Bo-June (Paul) Hsu and James Glass. 2006. Style & topic language model adaptation using HMM-LDA. In *Proc. Empirical Methods in Natural Language Processing*, pages 373–381, Sydney, Australia.

Bo-June (Paul) Hsu and James Glass. 2008. Iterative language model estimation: Efficient data structure & algorithms. In *Proc. Interspeech*, Brisbane, Australia.

Bo-June (Paul) Hsu. 2007. Generalized linear interpolation of language models. In *Proc. Automatic Speech Recognition and Understanding*, pages 136–140, Kyoto, Japan.

Igor Malioutov and Regina Barzilay. 2006. Minimum cut model for spoken lecture segmentation. In *Proc. Association for Computational Linguistics*, pages 25–32, Sydney, Australia.

D. Pallet, W. Fisher, and Fiscus. 1990. Tools for the analysis of benchmark speech recognition tests. In *Proc. ICASSP*, Albuquerque, NM, USA.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 2007. *Numerical Recipes*. Cambridge University Press, 3rd edition.

Michael Steinbach, George Karypis, and Vipin Kumar. 2000. A comparison of document clustering techniques. Technical Report #00-034, University of Minnesota.

Complexity of finding the BLEU-optimal hypothesis in a confusion network

Gregor Leusch and **Evgeny Matusov** and **Hermann Ney**
RWTH Aachen University, Germany
{leusch,matusov,ney}@cs.rwth-aachen.de

Abstract

Confusion networks are a simple representation of multiple speech recognition or translation hypotheses in a machine translation system. A typical operation on a confusion network is to find the path which minimizes or maximizes a certain evaluation metric. In this article, we show that this problem is generally NP-hard for the popular BLEU metric, as well as for smaller variants of BLEU. This also holds for more complex representations like generic word graphs. In addition, we give an efficient polynomial-time algorithm to calculate unigram BLEU on confusion networks, but show that even small generalizations of this data structure render the problem to be NP-hard again.

Since finding the optimal solution is thus not always feasible, we introduce an approximating algorithm based on a multi-stack decoder, which finds a (not necessarily optimal) solution for n-gram BLEU in polynomial time.

1 Introduction

In machine translation (MT), confusion networks (CNs) are commonly used to represent alternative versions of sentences. Typical applications include translation of different speech recognition hypotheses (Bertoldi et al., 2007) or system combination (Fiscus, 1997; Matusov et al., 2006).

A typical operation on a given CN is to find the path which minimizes or maximizes a certain evaluation metric. This operation can be used in applications like Minimum Error Rate Training (Och, 2003), or optimizing system combination as described by Hillard et al. (2007). Whereas this is easily achievable for simple metrics like the *Word Error Rate* (WER) as described by Mohri and Riley (2002), current research in MT uses more sophisticated measures, like the BLEU score (Papineni et al., 2001). Zens and Ney (2005) first described this task on general word graphs, and sketched a complete algorithm for calculating the maximum BLEU score in a word graph. While they do not give an estimate on the complexity of their algorithm, they note that already a simpler algorithm for calculating the Position independent Error Rate (PER) has an exponential worst-case complexity. The same can be expected for their BLEU algorithm. Dreyer et al (2007) examined a special class of word graphs, namely those that denote constrained reorderings of single sentences. These word graphs have some properties which simplify the calculation; for example, no edge is labeled with the empty word, and all paths have the same length and end in the same node. Even then, their decoder does not optimize the true BLEU score, but an approximate version which uses a language-model-like unmodified precision. We give a very short introduction to CNs and the BLEU score in Section 2.

In Section 3 we show that finding the best BLEU score is an NP-hard problem, even for a simplified variant of BLEU which only scores unigrams and bigrams. The main reason for this problem to become NP-hard is that by looking at bigrams, we allow for one decision to also influence the following decision, which itself can influence the decisions after that. We also show that this also holds for unigram BLEU and the *position independent error rate* (PER) on a slightly augmented variant of CNs which allows for edges to carry multiple symbols. The concatenation of symbols corresponds to the interdependency of decisions in the case of bigram matches above.

NP-hard problems are quite common in machine translation; for example, Knight (1999) has shown that even for a simple form of statistical MT models, the decoding problem is NP-complete. More recently, DeNero and Klein (2008) have proven the NP-completeness of the phrase alignment problem.

But even a simple, common procedure as BLEU scoring, which can be performed in linear time on single sentences, becomes a potentially intractable problem as soon as it has to be performed on a slightly more powerful representation, such as confusion networks. This rather surprising result is the motivation of this paper.

The problem of finding the best unigram BLEU score in an unaugmented variant of CNs is not NP-complete, as we show in Section 4. We present an algorithm that finds such a unigram BLEU-best path in polynomial time.

An important corollary of this work is that calculating the BLEU-best path on general word graphs is also NP-complete, as CNs are a true subclass of word graphs. It is still desirable to calculate a "good" path in terms of the BLEU score in a CN, even if calculating the best path is infeasible. In Section 5, we present an algorithm which can calculate "good" solutions for CNs in polynomial time. This algorithm can easily be extended to handle arbitrary word graphs. We assess the algorithm experimentally on real-world MT data in Section 6, and draw some conclusions from the results in this article in Section 7.

2 Confusion networks

A *confusion network* (CN) is a word graph where each edge is labeled with exactly zero or one symbol, and each path from the start node to the end node visits each node of the graph in canonical order. Usually, we represent unlabeled edges by labeling them with the *empty word* ε.

Within this paper, we represent a CN by a list of lists of words $\{w_{i,j}\}$, where each $w_{i,j}$ corresponds to a symbol on an edge between nodes i and $i+1$. A *path* in this CN can be written as a string of integers, $a_1^n = a_1, \ldots, a_n$, such that the path is labeled $w_{1,a_1} w_{2,a_2} \ldots w_{n,a_n}$. Note that there can be a different number of possible words, j, for different positions i.

2.1 BLEU and variants

The BLEU score, as defined by Papineni et al. (2001), is the modified n-gram precision of a hypothesis, with $1 \leq n \leq N$, given a set of reference translations R. "Modified precision" here means that for each n-gram, its maximum number of occurrences within the reference sentences is counted, and only up to that many occurrences in the hypothesis are considered to be correct. The geometric mean over the precisions for all n is calculated, and multiplied by a *brevity penalty* bp. This brevity penalty is 1.0 if the hypothesis sentence is at least as long as the reference sentence (special cases occur if multiple reference sentences with different length exists), and less than 1.0 otherwise. The exact formulation can be found in the cited paper; for the proofs in our paper it is enough to note that the BLEU score is 1.0 exactly if all n-grams in the hypothesis occur at least that many times in a reference sentence, and if there is a reference sentence which is as long as or shorter than the hypothesis. Assuming that we can always provide a dummy reference sentence shorter than this length, we do not need to regard the brevity penalty in these proofs. Within the following proofs of NP-hardness, we will only require confusion networks (and word graphs) which do not contain empty words, and where all paths from the start node to the end node have the same length.

Usually, in the definition of the BLEU score, N is set to 4; within this article we denote this metric as 4BLEU. We can also restrict the calculations to unigrams only, which would be 1BLEU, or to bigrams and unigrams, which we denote as 2BLEU.

Similar to the 1BLEU metric is the Position independent Error Rate PER (Tillmann et al., 1997), which counts the number of substitutions, insertions, and deletions that have to be performed on the unigram counts to have the hypothesis counts match the reference counts. Unlike 1BLEU, for PER to be optimal (here, 0.0), the reference counts must match the candidate counts exactly.

Given a CN $\{w_{i,j}\}$ and a set of reference sentences R, we define the optimization problem

Definition 1 (CN-2BLEU-OPTIMIZE) *Among all paths a_1^I through the CN, what is the path with the highest 2BLEU score?*

Related to this is the decision problem

Definition 2 (CN-2BLEU-DECIDE) *Among all paths a_1^I through the CN, is there a path with a 2BLEU score of* 1.0*?*

Similarly we define CN-4BLEU-DECIDE, CN-PER-DECIDE, etc.

3 CN-2BLEU-DECIDE is NP-complete

We now show that CN-2BLEU-DECIDE is NP-complete. It is obvious that the problem is in NP: Given a path a_1^I, which is polynomial in size to the problem, we can decide in polynomial time whether a_1^I is a solution to the problem – namely by calculating the BLEU score. We now show that there is a problem known to be NP-complete which can be polynomially reduced to CN-2BLEU-DECIDE. For our proof, we choose 3SAT.

3.1 3SAT

Consider the following problem:

Definition 3 (3SAT) *Let $X = \{x_1, \dots, x_n\}$ be a set of Boolean variables, let $\mathcal{F} = \bigwedge_{i=1}^{k} (L_{i,1} \vee L_{i,2} \vee L_{i,3})$ be a Boolean formula, where each literal $L_{i,j}$ is either a variable x or its negate $\overline{x}$. Is there a assignment $\beta : X \to \{0, 1\}$ such that $\beta \models \mathcal{F}$? In other words, if we replace each x in $\mathcal{F}$ by $\beta(x)$, and each $\overline{x}$ by $1 - \beta(x)$, does $\mathcal{F}$ become true?*

It has been shown by Karp (1972) that 3SAT is NP-complete. Consequently, if for another problem in NP there is polynomial-size and -time reduction of an arbitrary instance of 3SAT to an instance of this new problem, this new problem is also NP-complete.

3.2 Reduction of 3SAT to CN-2BLEU-DECIDE

Let $\mathcal{F}$ be a Boolean formula in 3CNF, and let k be its size, as in Definition 3. We will now reduce it to a corresponding CN-2BLEU-DECIDE problem. This means that we create an alphabet Σ, a confusion network $\mathcal{C}$, and a set of reference sentences R, such that there is a path through $\mathcal{C}$ with a BLEU score of 1.0 exactly if $\mathcal{F}$ is solvable:

Create an alphabet Σ based on $\mathcal{F}$ as $\Sigma := \{x_1, \dots, x_n\} \cup \{\overline{x_1}, \dots, \overline{x_n}\} \cup \{\diamond\}$. Here, the x_i and $\overline{x_i}$ symbols will correspond to the variable with the same name or their negate, respectively, whereas $\diamond$ will serve as an "isolator symbol", to avoid unwanted bigram matches or mismatches between separate parts of the constructed CN or sentences.

Consider the CN $\mathcal{C}$ from Figure 1.

Consider the following set of reference sentences:

$$\begin{aligned} R := \{ & \diamond (x_1\diamond)^k (x_2\diamond)^k \dots (x_n\diamond)^k \\ & \diamond (\overline{x_1}\diamond)^k (\overline{x_2}\diamond)^k \dots (\overline{x_n}\diamond)^k, \\ & (x_1)^k \diamond (\overline{x_1})^k \diamond \dots (x_n)^k \diamond (\overline{x_n})^k \diamond (\diamond)^{k+n} \ \} \end{aligned}$$

where $(x)^k$ denotes k subsequent occurrences of x. Clearly, both $\mathcal{C}$ and R are of polynomial size in n and k, and can be constructed in polynomial time.

Then,

There is an assignment β such that $\beta \models \mathcal{F}$

$\Leftrightarrow$

There is a path a_1^I through $\mathcal{C}$ such that $\mathrm{BLEU}(a_1^I, R) = 1.0$.

Proof: "$\Rightarrow$"

Let β be an assignment under which $\mathcal{F}$ becomes true. Create a path a_1^I as follows: Within $\mathcal{A}$, for each set of edges $L_{i,1}$, $L_{i,2}$, $L_{i,3}$, choose the path through an x where $\beta(x) = 1$, or through an $\overline{x}$ where $\beta(x) = 0$. Note that there must be such an x, because otherwise the clause $L_{i,1} \vee L_{i,2} \vee L_{i,3}$ would not be true under β. Within $\mathcal{B}$, select the path always through x_i if $\beta(x_i) = 0$, and through $\overline{x_i}$ if $\beta(x_i) = 1$.

Then, a_1^I consists of, for each i,

- At most k occurrences of both x_i and $\overline{x_i}$
- At most k occurrences of each of the bigrams $x_i\diamond$, $\diamond x_i$, $\overline{x_i}\diamond$, $\diamond\overline{x_i}$, $x_i x_i$, and $\overline{x_i}\,\overline{x_i}$
- No other bigram than those listed above.

For all of these unigram and bigram counts, there is a reference sentence in R which contains at least as many of those unigrams/bigrams as the path. Thus, the unigram and bigram precision of a_1^I is 1.0. In addition, there is always a reference sentence whose length is shorter than that of a_1^I, such that the brevity penalty is also 1.0. As a result, $\mathrm{BLEU}(a_1^I, R) = 1.0$.

"$\Leftarrow$"

Let a_1^I be a path through $\mathcal{C}$ such that $BLEU(a_1^I, R) = 1.0$. Because there is no bigram $x_i\overline{x_i}$ or $\overline{x_i}x_i$ in R, we can assume that for each x_i, either only x_i edges, or only $\overline{x_i}$ edges appear in the $\mathcal{B}$ part of a_1^I, each at most k times. As no unigram x_i and $\overline{x_i}$ appears more than k times in R, we can assume that, if the $\overline{x_i}$ edges are passed in $\mathcal{B}$, then only the x_i edges are passed in $\mathcal{A}$, and vice versa. Now, create an assignment β as follows:

$$\beta := \begin{cases} 0 & \text{if } x_i \text{ edges are passed in } \mathcal{B} \\ 1 & \text{otherwise} \end{cases}$$

Then, $\beta \models \mathcal{F}$. Proof: Assume that $\mathcal{F}_\beta = 0$. Then there must be a clause i such that $L_{i,1} \vee L_{i,2} \vee L_{i,3} =$

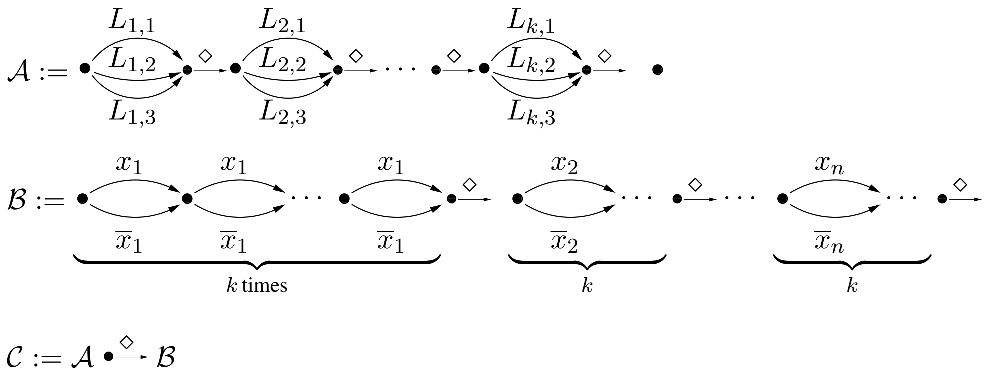


Figure 1: CN constructed from a 3SAT formula $\mathcal{F}$. $\mathcal{C}$ is the concatenation of the left part $\mathcal{A}$, and the right path $\mathcal{B}$, separated by an isolating $\diamond$.

0. At least one of the edges $L_{i,j}$ associated with the literals of this clause must have been passed by a_1^K in $\mathcal{A}$. This literal, though, can not have been passed in $\mathcal{B}$. As a consequence, $\beta(L_{i,j}) = 1$. But this means that $L_{i,1} \vee L_{i,2} \vee L_{i,3} = 1 \quad \rightsquigarrow$ contradiction.

Because CN-2BLEU-DECIDE is in NP, and we can reduce an NP-complete problem (3SAT) in polynomial time to a CN-2BLEU-DECIDE problem, this means that CN-2BLEU-DECIDE is NP-complete.

3.3 CN-4BLEU-DECIDE

It is straightforward to modify the construction above to create an equivalent CN-4BLEU-DECIDE problem instead: Replace each occurrence of the isolating symbol $\diamond$ in $\mathcal{A}, \mathcal{B}, \mathcal{C}, R$ by three consecutive isolating symbols $\diamond\diamond\diamond$. Then, everything said about unigrams still holds, and bi-, tri- and fourgrams are handled equivalently: Previous unigram matches on $\diamond$ correspond to uni-, bi-, and trigram matches on $\diamond$, $\diamond\diamond$, $\diamond\diamond\diamond$. Bigram matches on $x\diamond$ correspond to bi-, tri-, and fourgram matches on $x\diamond$, $x\diamond\diamond$, $x\diamond\diamond\diamond$, and similar holds for bigram matches $\overline{x}\diamond$, $\diamond x$, $\diamond\overline{x}$. Unigram matches x, $\overline{x}$, and bigram matches xx etc. stay the same. Consequently, CN-4BLEU-DECIDE is also an NP-complete problem.

3.4 CN*-1BLEU-DECIDE

Is it possible to get rid of the necessity for bigram counts in this proof? One possibility might be to look at slightly more powerful graph structures, CN*. In these graphs, each edge can be labeled by arbitrarily many symbols (instead of just zero or one). Then, consider a CN* graph $\mathcal{C}' := \mathcal{A}\bullet\xrightarrow{\diamond}\mathcal{B}'$, with $\mathcal{B}'$ as in Figure 2.

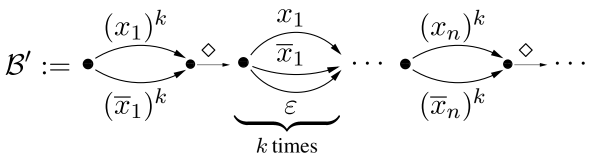


Figure 2: Right part of a CN* constructed from a 3SAT formula $\mathcal{F}$.

With

$$R' := \{(x_1)^k(\overline{x_1})^k \dots (x_n)^k(\overline{x_n})^k(\diamond)^k\}$$

we can again assume that either x_i or $\overline{x_i}$ appears k times in the $\mathcal{B}'$-part of a path a_1^K with $1BLEU(a_1^K, R') = 1.0$, and that for every solution β to $\mathcal{F}$ there is a corresponding path a_1^K through $\mathcal{C}'$ and vice versa. In this construction, we also have exact matches of the counts, so we can also use PER in the decision problem.

While CN* are generally not word graphs by themselves due to the multiple symbols on edges, it is straightforward to create an equivalent word graph from a given CN*, as demonstrated in Figure 3. Consequently, deciding unigram BLEU and unigram PER are NP-complete problems for general word graphs as well.

4 Solving CN-1BLEU-DECIDE in polynomial time

It is not a coincidence that we had to resort to bigrams or to edges with multiple symbols for NP-completeness: It turns out that CN-1BLEU-DECIDE, where the order of the words does not

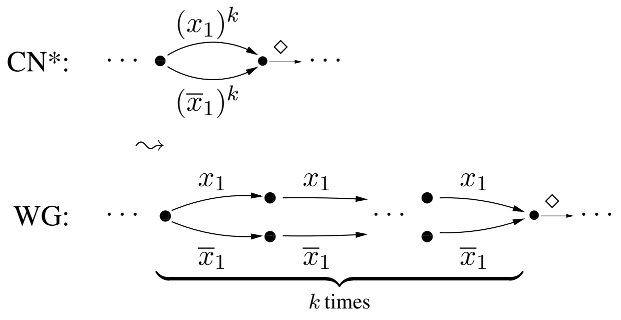
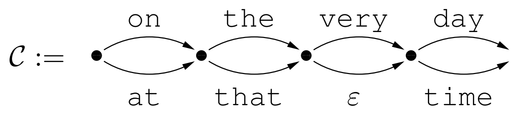


Figure 3: Construction of a word graph from a CN* as in $\mathcal{B}'$.

matter at all, can be decided in polynomial time using the following algorithm, which disregards a brevity penalty for the sake of simplicity:

Given a vocabulary X, a CN $\{w_{i,j}\}$, and a set of reference sentences R together with their unigram BLEU counts $c(x) : X \to \mathbb{N}$ and $C := \sum_{x \in X} c(x)$,

1. Remove all parts from w where there is an edge labeled with the empty word ε. This step will always increase unigram precision, and can not hurt any higher n-gram precision here, because $n = 1$. In the example in Figure 4, the edges labeled `very` and ε respectively are affected in this step.
2. Create nodes $A_0 := \{1, \dots, n\}$, one for each node with edges in the CN. In the example in Figure 5, the three leftmost column heads correspond to these nodes.
3. Create nodes $B := \{x.j \mid x \in X, 1 \le j \le c(x)\}$. In other words, create a unique node for each "running" word in R – e.g. if the first and second reference sentence contain x once each, and the third reference contains x twice, create exactly $x.1$ and $x.2$. In Figure 5, those are the row heads to the right.
4. Fill A with empty nodes to match the total length: $A := A_0 \cup \{\varepsilon.j \mid 1 \le j \le C - n\}$.
   If $n > C$, the BLEU precision can not be 1.0. The five rightmost columns in Figure 5 correspond to those.
5. Create edges
   $E := \{(i, w_{i,j}.k) \mid 1 \le i \le n, \text{ all } j, 1 \le c(w_{i,j})\}$ $\cup \{(i, \varepsilon.j) \mid 1 \le i \le n, \text{ all } j\}$. These edges are denoted as ○ or ● in Figure 5.



$R := \{$ `on the same day,`
`at the time and the day` $\}$

Figure 4: Example for CN-1BLEU-DECIDE.

1	2	3	$\varepsilon.1$	$\varepsilon.2$	$\varepsilon.3$	$\varepsilon.4$	$\varepsilon.5$	
●			○	○	○	○	○	`on`
	●		○	○	○	○	○	`the.1`
	○		●	○	○	○	○	`the.2`
			○	●	○	○	○	`same`
		●	○	○	○	○	○	`day`
○			○	○	●	○	○	`at`
		○	○	○	○	●	○	`time`
			○	○	○	○	●	`and`

Figure 5: Bipartite graph constructed to find the optimal 1BLEU path in Figure 4. One possible maximum bipartite matching is marked with ●.

6. Find the *maximum bipartite matching* M between A and B given E. Figure 5 shows such a matching with ●.
7. If all nodes in A and B are covered by M, then 1BLEU$(\{w_{i,j}\}, R) = 1.0$. The words that are matched to A_0 then form the solution path through $\{w_{i,j}\}$.

Figure 4 gives an example of a CN and a set of references R, for which the best 1BLEU path can be constructed by the algorithm above. The bipartite graph constructed in Step 1 to Step 4 for this example, given in matrix form, can be found in Figure 5.

Such a solution to Step 6, if found, corresponds exactly to a path through the confusion network with 1BLEU=1.0, and vice versa: for each position $1 \le i \le n$, the matched word corresponds to the word that is selected for the position of the path; "surplus" counts are matched with εs.

Step 6 can be performed in polynomial time (Hopcroft and Karp, 1973) $O((C+n)^{5/2})$; all other steps in linear time $O(C+n)$. Consequently, CN-1BLEU can be decided in polynomial time $O((C+n)^{5/2})$. Similarly, an actual optimum 1BLEU score can be calculated in $O((C+n)^{5/2})$.

It should be noted that the only alterations in the hypothesis length, and as a result the only alterations in the brevity penalty, will come from Step 1. Consequently, the brevity penalty can be taken into account as follows: Consider that there are M nodes with an empty edge in $\{w_{i,j}\}$. Instead of removing them in Step 1, keep them in, but for each $1 \leq m \leq M$, run through steps 2 to 6, but add m nodes $\varepsilon.1, \ldots, \varepsilon.m$ to B in Step 3, and add corresponding edges to these nodes to E in Step 5. After each iteration (which leads to a constant hypothesis length), calculate precision and brevity penalty. Select the best product of precision and brevity penalty in the end. The overall time complexity now is in $M \cdot O((C+n)^{5/2})$.

A PER score can be calculated in a similar fashion.

5 Finding approximating solutions for CN-4BLEU in polynomial time

Knowing that the problem of finding the BLEU-best path is an NP-complete problem is an unsatisfactory answer in practice – in many cases, having a good, but not necessarily optimum path is preferable to having no good path at all.

A simple approach would be to walk the CN from the start node to the end node, keeping track of n-grams visited so far, and choosing the word next which maximizes the n-gram precision up to this word. Track is kept by keeping n-gram count vectors for the hypothesis path and the reference sentences, and update those in each step.

The main problem with this approach is that often the local optimum is suboptimal on the global scale, for example if a word occurs on a later position again.

Zens and Ney (2005) on the other hand propose to keep all n-gram count vectors instead, and only recombine path hypotheses with identical count vectors. As they suspect, the search space can become exponentially large.

In this paper, we suggest a compromise between these two extremes, namely keeping active a sufficiently large number of "path hypotheses" in terms of n-gram precision, instead of only the first best, or of all. But even then, edges with empty words pose a problem, as stepping along an empty edge will never decrease the precision of the local path. In certain cases, steps along empty edges may affect the n-gram precision for higher n-grams. But this will only take effect after the next non-empty step, it does not influence the local decision in a node. Stepping along a non-empty edge will often decrease the local precision, though. As a consequence, a simple algorithm will prefer paths with shorter hypotheses, which leads to a suboptimal total BLEU score, because of the brevity penalty. One can counter this problem for example by using a brevity penalty already during the search. But this is problematic as well, because it is difficult to define a proper partial reference length in this case.

The approach we propose is to compare only partial path hypotheses with the same number of empty edges, and ending in the same position in the confusion network. This idea is illustrated in Figure 6: We compare only the partial precision of path hypotheses ending in the same node. Due to the simple nature of this search graph, it can easily be traversed in a left-to-right, top-to-bottom manner. With regard to a node currently being expanded, only the next node in the same row, and the corresponding columns in the next row need to be kept active. When implementing this algorithm, Hypotheses should be compared on the modified BLEUS precision by Lin and Och (2004) because the original BLEU precision equals zero as long as there are no higher n-gram matches in the partial hypotheses, which renders meaningful comparison hard or impossible.

In the rightmost column, all path hypotheses within a node have the same hypothesis length. Consequently, we can select the hypothesis with the best (brevity-penalized) BLEU score by multiplying the appropriate brevity penalty to the precision of the best path ending in each of these nodes. If we always expand all possible path hypotheses within the nodes, and basically run a full search, we will always find the BLEU-best path this way. From the proof above, it follows that the number of path hypothesis we would have to keep can become exponentially large. Fortunately, if a "good" solution is good enough, we do not have to keep all possible path hypotheses, but only the S best ones for a given constant S, or those with a precision not worse than c times the precision of the best hypothesis within the node. Assuming that adding and removing an element to/from a size-limited stack of size S takes time $O(\log S)$, that we allow at most E empty edges in a solution, and that there are j edges in each of the n positions, this algorithm has a time complexity of

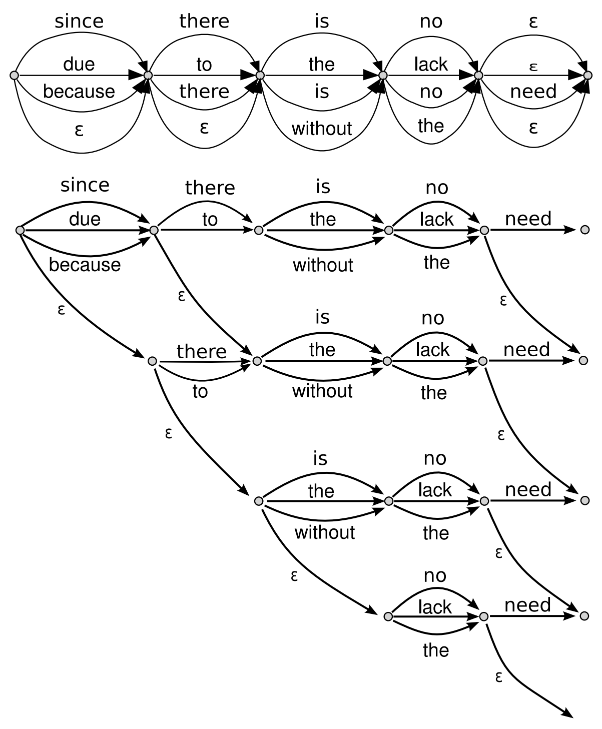


Figure 6: Principle of the multi-stack decoder used to find a path with a good BLEU score. The first row shows the original confusion network, the following rows show the search graph. Duplicate edges were removed, but no word was considered "unknown".

$O(E \cdot n \cdot j \cdot S \log S)$.

To reduce redundant duplicated path hypotheses, and by this to speed up the algorithm and reduce the risk that good path hypotheses are pruned, the confusion network should be *simplified* before the search, as shown in Figure 6:

1. Remove all words in the CN which do not appear in any reference sentence, if there at least one "known" non-empty word at the same position. If there is no such "known" word, replace them all by a single token denoting the "unknown word".

2. Remove all duplicate edges in a position, that is, if there are two or more edges carrying the same label in one position, remove all but one for them.

These two steps will keep at least one of the BLEU-best paths intact. But they can remove the average branching factor (j) of the CN significantly, which leads to a significantly lower number of duplicate path hypotheses during the search.

Table 1: Statistics of the (Chinese–)English MT corpora used for the experiments

	NIST 2003	NIST 2006	
number of systems	4	4	
number of ref.	4	4	per sent.
sentences	919	249	
system length	28.4	33.2	words*
ref. length	27.5	34.2	words*
best path	24.4	33.9	words*
CN length	40.7	39.5	nodes*
best single system	29.3	52.5	BLEU
	30.5	51.6	BLEUS*

*average per sentence

Our algorithm can easily be extended to handle arbitrary word graphs instead of confusion networks. In this case, each "row" in Figure 6 will reflect the structure of the word graph instead of the "linear" structure of the CN.

While this algorithm searches for the best path for a single sentence only, a common task is to find the best BLEU score over a whole test set – which can mean suboptimal BLEU scores for individual sentences. This adds an additional combinatorial problem over the sentences to the actual decoding process. Both Zens and Ney (2005) and Dreyer et al (2007) use a greedy approach here; the latter estimated the impact of this to be insignificant in random sampling experiments. In our experiments, we used the per-sentence BLEUS score as (greedy) decision criterion, as this is also the pruning criterion. One possibility to adapt this approach to Zens's/Dreyer's greedy approach for system-level BLEU scores might be to initialize n-gram counts and hypothesis length not to zero at the beginning of each sentence, but to those of the corpus so far. But as this diverts from our goal to optimize the sentence-level scores, we have not implemented it so far.

6 Experimental assessment of the algorithm

The question arises how many path hypotheses we need to retain in each step to obtain optimal paths. To examine this, we created confusion networks out of the translations of the four best MT systems of

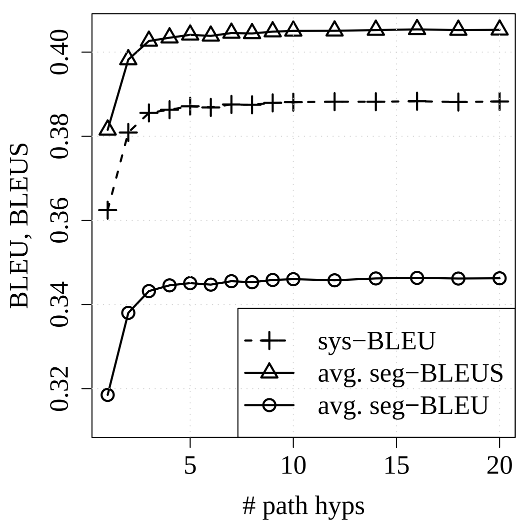


Figure 7: Average of the sentence-wise BLEU and BLEUS score and the system-wide BLEU score versus the number of path hypotheses kept per node during the search. NIST MT03 corpus.

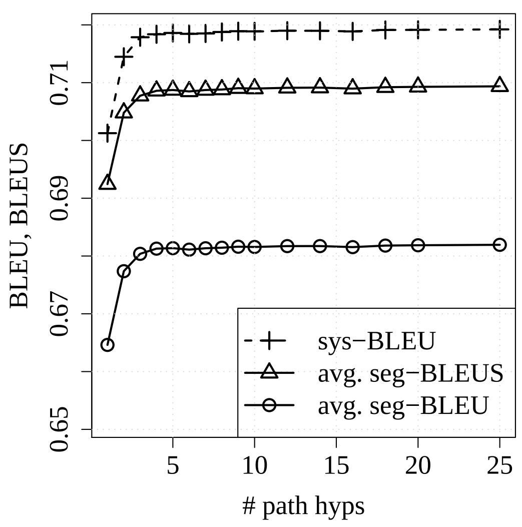


Figure 8: Average of the sentence-wise BLEU and BLEUS score and the system-wide BLEU score versus the number of path hypotheses kept per node during the search. NIST MT06 corpus.

the NIST 2003 and 2006 Chinese–English evaluation campaigns, as available from the Linguistic Data Consortium (LDC). The hypotheses of the best single system served as skeleton, those of the three remaining systems were reordered and aligned to the skeleton hypothesis. This corpus is described in Table 1. Figures 7 and 8 show the measured BLEU scores in three different definitions, versus the maximum number of path hypotheses that are kept in each node of the search graph. Shown are the average sentence-wise BLEUS score, which is what the algorithm actually optimizes, for comparison the average sentence-wise BLEU score, and the total document-wise BLEU score.

All scores increase with increasing number of retained hypotheses, but stabilize around a total of 15 hypotheses per node. The difference over a greedy approach, which corresponds to a maximum of one hypothesis per node if we leave out the separation by path length, is quite significant. No further improvements can be expected for a higher number of hypotheses, as experiments up to 100 hypotheses show.

7 Conclusions

In this paper, we showed that deciding whether a given CN contains a path with a BLEU score of 1.0 is an NP-complete problem for n-gram lengths ≥ 2. The problem is also NP-complete if we only look at unigram BLEU, but allow for CNs where edges may contain multiple symbols, or for arbitrary word graphs. As a corollary, any proposed algorithm to find the path with an optimal BLEU score in a CN, even more in an arbitrary word graph, which runs in worst case polynomial time can only deliver an approximation[1].

We gave an efficient polynomial time algorithm for the simplest variant, namely deciding on a unigram BLEU score for a CN. This algorithm can easily be modified to decide on the PER score as well, or to calculate an actual unigram BLEU score for the hypothesis CN.

Comparing these results, we conclude that the ability to take bi- or higher n-grams into account, be it in the scoring (as in 2BLEU), or in the graph structure (as in CN*), is the key to render the problem NP-hard. Doing so creates long-range dependencies, which oppose local decisions.

We also gave an efficient approximating algorithm for higher-order BLEU scores. This algorithm is based on a multi-stack decoder, taking into account the empty arcs within a path. Experimental results on real-world data show that our method is indeed able to find paths with a significantly better

[1]provided that $P \neq NP$, of course.

BLEU score than that of a greedy search. The resulting BLEUS score stabilizes already on a quite restricted search space, showing that despite the proven NP-hardness of the exact problem, our algorithm can give useful approximations in reasonable time. It is yet an open problem in how far the problems of finding the best paths regarding a sentence-level BLEU score, and regarding a system-level BLEU score correlate. Our experiments here suggest a good correspondence.

8 Acknowledgments

This paper is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023. The proofs and algorithms in this paper emerged while the first author was visiting researcher at the Interactive Language Technologies Group of the National Research Council (NRC) of Canada, Gatineau. The author wishes to thank NRC and Aachen University for the opportunity to jointly work on this project.

References

Nicola Bertoldi, Richard Zens, and Marcello Federico. 2007. Speech translation by confusion network decoding. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1297–1300, Honululu, HI, USA, April.

John DeNero and Dan Klein. 2008. The complexity of phrase alignment problems. In *Human Language Technologies 2008: The Conference of the Association for Computational Linguistics, Short Papers*, pages 25–28, Columbus, Ohio, June. Association for Computational Linguistics.

Markus Dreyer, Keith Hall, and Sanjeev Khudanpur. 2007. Comparing Reordering Constraints for SMT Using Efficient BLEU Oracle Computation. In *AMTA Workshop on Syntax and Structure in Statistical Translation (SSST) at the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 103–110, Rochester, NY, USA, April.

Jonathan G. Fiscus. 1997. A post-processing system to yield reduced word error rates: Recogniser output voting error reduction (ROVER). In *Proceedings 1997 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 347–352, Santa Barbara, CA.

Dustin Hillard, Björn Hoffmeister, Mari Ostendorf, Ralf Schlüter, and Hermann Ney. 2007. iROVER: Improving system combination with classification. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 65–68, Rochester, New York, April.

John E. Hopcroft and Richard M. Karp. 1973. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231.

Richard M. Karp. 1972. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press.

Kevin Knight. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615, December.

Chin-Yew Lin and Franz Josef Och. 2004. Orange: a method for evaluation automatic evaluation metrics for machine translation. In *Proc. COLING 2004*, pages 501–507, Geneva, Switzerland, August.

Evgeny Matusov, Nicola Ueffing, and Hermann Ney. 2006. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. In *Conference of the European Chapter of the Association for Computational Linguistics*, pages 33–40, Trento, Italy, April.

Mehryar Mohri and Michael Riley. 2002. An efficient algorithm for the n-best-strings problem. In *Proc. of the 7th Int. Conf. on Spoken Language Processing (ICSLP'02)*, pages 1313–1316, Denver, CO, September.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of the 41th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 160–167, Sapporo, Japan, July.

Kishore A. Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. Technical Report RC22176 (W0109-022), IBM Research Division, Thomas J. Watson Research Center, September.

Christoph Tillmann, Stephan Vogel, Hermann Ney, Alex Zubiaga, and Hassan Sawaf. 1997. Accelerated DP based search for statistical translation. In *European Conf. on Speech Communication and Technology*, pages 2667–2670, Rhodes, Greece, September.

Richard Zens and Hermann Ney. 2005. Word graphs for statistical machine translation. In *43rd Annual Meeting of the Assoc. for Computational Linguistics: Proc. Workshop on Building and Using Parallel Texts: Data-Driven Machine Translation and Beyond*, pages 191–198, Ann Arbor, MI, June.

A Simple and Effective Hierarchical Phrase Reordering Model

Michel Galley
Computer Science Department
Stanford University
Stanford, CA 94305-9020
galley@cs.stanford.edu

Christopher D. Manning
Computer Science Department
Stanford University
Stanford, CA 94305-9010
manning@cs.stanford.edu

Abstract

While phrase-based statistical machine translation systems currently deliver state-of-the-art performance, they remain weak on word order changes. Current phrase reordering models can properly handle swaps between adjacent phrases, but they typically lack the ability to perform the kind of long-distance re-orderings possible with syntax-based systems. In this paper, we present a novel hierarchical phrase reordering model aimed at improving non-local reorderings, which seamlessly integrates with a standard phrase-based system with little loss of computational efficiency. We show that this model can successfully handle the key examples often used to motivate syntax-based systems, such as the rotation of a prepositional phrase around a noun phrase. We contrast our model with reordering models commonly used in phrase-based systems, and show that our approach provides statistically significant BLEU point gains for two language pairs: Chinese-English (+0.53 on MT05 and +0.71 on MT08) and Arabic-English (+0.55 on MT05).

1 Introduction

Statistical phrase-based systems (Och and Ney, 2004; Koehn et al., 2003) have consistently delivered state-of-the-art performance in recent machine translation evaluations, yet these systems remain weak at handling word order changes. The re-ordering models used in the original phrase-based systems penalize phrase displacements proportionally to the amount of nonmonotonicity, with no consideration of the fact that some words are far more likely to be displaced than others (e.g., in English-to-Japanese translation, a verb should typically move to the end of the clause).

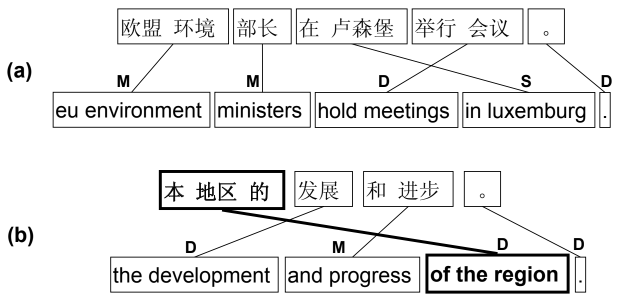


Figure 1: Phase orientations (monotone, swap, discontinuous) for Chinese-to-English translation. While previous work reasonably models phrase reordering in simple examples (a), it fails to capture more complex reorderings, such as the swapping of "of the region" (b).

Recent efforts (Tillman, 2004; Och et al., 2004; Koehn et al., 2007) have directly addressed this issue by introducing lexicalized reordering models into phrase-based systems, which condition reordering probabilities on the words of each phrase pair. These models distinguish three orientations with respect to the previous phrase—monotone (*M*), swap (*S*), and discontinuous (*D*)—and as such are primarily designed to handle *local* re-orderings of neighboring phrases. Fig. 1(a) is an example where such a model effectively swaps the prepositional phrase *in Luxembourg* with a verb phrase, and where the noun *ministers* remains in monotone order with respect to the previous phrase *EU environment*.

While these lexicalized re-ordering models have shown substantial improvements over unlexicalized phrase-based systems, these models only have a limited ability to capture sensible long distance re-orderings, as can be seen in Fig. 1(b). The phrase *of the region* should swap with the rest of the noun phrase, yet these previous approaches are unable to model this movement, and assume the orientation of this phrase is discontinuous (*D*). Observe that, in a shortened version of the same sentence (without *and progress*), the phrase orientation would be different (*S*), even though the shortened version has essentially the same sentence structure. Coming from the other direction, such observations about phrase reordering between different languages are precisely the kinds of facts that parsing approaches to machine translation are designed to handle and do successfully handle (Wu, 1997; Melamed, 2003; Chiang, 2005).

In this paper, we introduce a novel orientation model for phrase-based systems that aims to better capture long distance dependencies, and that presents a solution to the problem illustrated in Fig. 1(b). In this example, our reordering model effectively treats the adjacent phrases *the development* and *and progress* as one single phrase, and the displacement of *of the region* with respect to this phrase can be treated as a swap. To be able identify that adjacent blocks (e.g., *the development* and *and progress*) can be merged into larger blocks, our model infers binary (non-linguistic) trees reminiscent of (Wu, 1997; Chiang, 2005). Crucially, our work distinguishes itself from previous hierarchical models in that it does not rely on any cubic-time parsing algorithms such as CKY (used in, e.g., (Chiang, 2005)) or the Earley algorithm (used in (Watanabe et al., 2006)). Since our reordering model does not attempt to resolve natural language ambiguities, we can effectively rely on (linear-time) shift-reduce parsing, which is done jointly with left-to-right phrase-based beam decoding and thus introduces no asymptotic change in running time. As such, the hierarchical model presented in this paper maintains all the effectiveness and speed advantages of statistical phrase-based systems, while being able to capture some key linguistic phenomena (presented later in this paper) which have motivated the development of parsing-based approaches. We also illustrate this with results that are significantly better than previous approaches, in particular the lexical reordering models of Moses, a widely used phrase-based SMT system (Koehn et al., 2007).

This paper is organized as follows: the training of lexicalized re-ordering models is described in Section 3. In Section 4, we describe how to combine shift-reduce parsing with left-to-right beam search phrase-based decoding with the same asymptotic running time as the original phrase-based decoder. We finally show in Section 6 that our approach yields results that are significantly better than previous approaches for two language pairs and different test sets.

2 Lexicalized Reordering Models

We compare our re-ordering model with related work (Tillman, 2004; Koehn et al., 2007) using a log-linear approach common to many state-of-the-art statistical machine translation systems (Och and Ney, 2004). Given an input sentence **f**, which is to be translated into a target sentence **e**, the decoder searches for the most probable translation $\hat{\mathbf{e}}$ according to the following decision rule:

$$\hat{\mathbf{e}} = \arg\max_{\mathbf{e}} \left\{ p(\mathbf{e}|\mathbf{f}) \right\} \quad (1)$$

$$= \arg\max_{\mathbf{e}} \left\{ \sum_{j=1}^{J} \lambda_j h_j(\mathbf{f},\mathbf{e}) \right\} \quad (2)$$

$h_j(\mathbf{f},\mathbf{e})$ are J arbitrary feature functions over sentence pairs. These features include lexicalized re-ordering models, which are parameterized as follows: given an input sentence **f**, a sequence of target-language phrases $\mathbf{e} = (\overline{e}_1,\ldots,\overline{e}_n)$ currently hypothesized by the decoder, and a phrase alignment $\mathbf{a} = (a_1,\ldots,a_n)$ that defines a source $\overline{f}_{a_i}$ for each translated phrase $\overline{e}_i$, these models estimate the probability of a sequence of orientations $\mathbf{o} = (o_1,\ldots,o_n)$

$$p(\mathbf{o}|\mathbf{e},\mathbf{f}) = \prod_{i=1}^{n} p(o_i|\overline{e}_i,\overline{f}_{a_i},a_{i-1},a_i), \quad (3)$$

where each o_i takes values over the set of possible orientations $\mathscr{O} = \{M,S,D\}$.[1] The probability is conditioned on both a_{i-1} and a_i to make sure that the label o_i is consistent with the phrase alignment. Specifically, probabilities in these models can be

[1]We note here that the parameterization and terminology in (Tillman, 2004) is slightly different. We purposely ignore these differences in order to enable a direct comparison between Tillman's, Moses', and our approach.

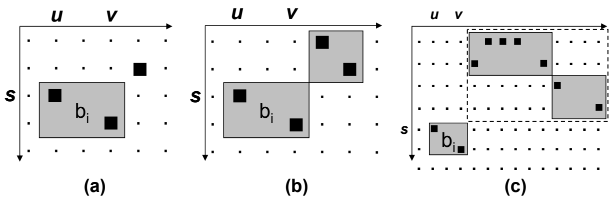


Figure 2: Occurrence of a swap according to the three orientation models: word-based, phrase-based, and hierarchical. Black squares represent word alignments, and gray squares represent blocks identified by phrase-extract. In (a), block $b_i = (e_i, f_{a_i})$ is recognized as a swap according to all three models. In (b), b_i is not recognized as a swap by the word-based model. In (c), b_i is recognized as a swap only by the hierarchical model.

greater than zero only if one of the following conditions is true:

- $o_i = M$ and $a_i - a_{i-1} = 1$
- $o_i = S$ and $a_i - a_{i-1} = -1$
- $o_i = D$ and $|a_i - a_{i-1}| \neq 1$

At decoding time, rather than using the log-probability of Eq. 3 as single feature function, we follow the approach of Moses, which is to assign three distinct parameters $(\lambda_m, \lambda_s, \lambda_d)$ for the three feature functions:

- $f_m = \sum_{i=1}^{n} \log p(o_i = M | \ldots)$
- $f_s = \sum_{i=1}^{n} \log p(o_i = S | \ldots)$
- $f_d = \sum_{i=1}^{n} \log p(o_i = D | \ldots)$.

There are two key differences between this work and previous orientation models (Tillman, 2004; Koehn et al., 2007): (1) the estimation of factors in Eq. 3 from data; (2) the segmentation of **e** and **f** into phrases, which is static in the case of (Tillman, 2004; Koehn et al., 2007), while it is dynamically updated with hierarchical phrases in our case. These differences are described in the two next sections.

3 Training

We present here three approaches for computing $p(o_i|\overline{e}_i, \overline{f}_{a_i}, a_{i-1}, a_i)$ on word-aligned data using relative frequency estimates. We assume here that phrase $\overline{e}_i$ spans the word range $s, \ldots, t$ in the target sentence **e** and that the phrase $\overline{f}_{a_i}$ spans the range

Orientation Model	$o_i = M$	$o_i = S$	$o_i = D$
word-based (Moses)	0.1750	0.0159	0.8092
phrase-based	0.3192	0.0704	0.6104
hierarchical	0.4878	0.1004	0.4116

Table 1: Class distributions of the three orientation models, estimated from 12M words of Chinese-English data using the grow-diag alignment symmetrization heuristic implemented in Moses, which is similar to the 'refined' heuristic of (Och and Ney, 2004).

$u, \ldots, v$ in the source sentence **f**. All phrase pairs in this paper are extracted with the phrase-extract algorithm (Och and Ney, 2004), with maximum length set to 7.

Word-based orientation model: This model analyzes word alignments at positions $(s-1, u-1)$ and $(s-1, v+1)$ in the alignment grid shown in Fig. 2(a). Specifically, orientation is set to $o_i = M$ if $(s-1, u-1)$ contains a word alignment and $(s-1, v+1)$ contains no word alignment. It is set to $o_i = S$ if $(s-1, u-1)$ contains no word alignment and $(s-1, v+1)$ contains a word alignment. In all other cases, it is set to $o_i = D$. This procedure is exactly the same as the one implemented in Moses.[2]

Phrase-based orientation model: The model presented in (Tillman, 2004) is similar to the word-based orientation model presented above, except that it analyzes adjacent phrases rather than specific word alignments to determine orientations. Specifically, orientation is set to $o_i = M$ if an adjacent phrase pair lies at $(s-1, u-1)$ in the alignment grid. It is set to S if an adjacent phrase pair covers $(s-1, v+1)$ (as shown in Fig. 2(b)), and is set to D otherwise.

Hierarchical orientation model: This model analyzes alignments beyond adjacent phrases. Specifically, orientation is set to $o_i = M$ if the phrase-extract algorithm is able to extract a phrase pair at $(s-1, u-1)$ given no constraint on maximum phrase length. Orientation is S if the same is true at $(s-1, v+1)$, and orientation is D otherwise.

Table 1 displays overall class distributions according to the three models. It appears clearly that occurrences of M and S are too sparsely seen in the word-based model, which assigns more than 80% of its

[2]http://www.statmt.org/moses/?n=Moses.AdvancedFeatures

			word	*phrase*	*hier.*
Monotone with previous			$p(o_i = M \mid \overline{e}_i, \overline{f}_{a_i}, a_{i-1}, a_i)$		
1	，是	and is	0.223	0.672	0.942
2	，也	and also	0.201	0.560	0.948
Swap with previous			$p(o_i = S \mid \overline{e}_i, \overline{f}_{a_i}, a_{i-1}, a_i)$		
3	中国 的	of china	0.303	0.617	0.651
4	他 说	, he said	0.003	0.030	0.395
Monotone with next			$p(o_i = M \mid \overline{e}_i, \overline{f}_{a_i}, a_{i+1}, a_i)$		
5	他 指出，	he pointed out that	0.601	0.770	0.991
6	然而，	however ,	0.517	0.728	0.968
Swap with next			$p(o_i = S \mid \overline{e}_i, \overline{f}_{a_i}, a_{i+1}, a_i)$		
7	的 发展	the development of	0.145	0.831	0.900
8	的 邀请	at the invitation of	0.272	0.834	0.925

Table 2: Monotone and swap probabilities for specific phrases according to the three models (word, phrase, and hierarchical). To ensure probabilities are representative, we only selected phrase pairs that occur at least 100 times in the training data.

probability mass to D. Conversely, the hierarchical model counts considerably less discontinuous cases, and is the only model that accounts for the fact that real data is predominantly monotone.

Since D is a rather uninformative default category that gives no clue how a particular phrase should be displaced, we will also provide MT evaluation scores (in Section 6) for a set of classes that distinguishes between left and right discontinuity $\{M, S, D_l, D_r\}$, a choice that is admittedly more linguistically motivated.

Table 2 displays orientation probabilities for concrete examples. Each example was put under one of the four categories that linguistically seems the best match, and we provide probabilities for that category according to each model. Note that, while we have so far only discussed left-to-right reordering models, it is also possible to build right-to-left models by substituting a_{i-1} with a_{i+1} in Eq. 3. Examples for right-to-left models appear in the second half of the table. The table strongly suggests that the hierarchical model more accurately determines the orientation of phrases with respect to large contextual blocks. In Examples 1 and 2, the hierarchical model captures the fact that coordinated clauses almost always remain in the same order, and that words should generally be forbidden to move from one side of "and" to the other side, a constraint that is difficult to enforce with the other two reordering models. In Example 4, the first two models completely ignore that "he said" sometimes rotates around its neighbor clause.

4 Decoding

Computing reordering scores during decoding with word-based[3] and phrase-based models (Tillman, 2004) is trivial, since they only make use of local information to determine the orientation of a new incoming block b_i. For a left-to-right ordering model, b_i is scored based on its orientation with respect to b_{i-1}. For instance, if b_i has a swap orientation with respect to the previous phrase in the current translation hypothesis, feature $p(o_i = S|\ldots)$ becomes active.

Computing lexicalized reordering scores with the hierarchical model is more complex, since the model must identify contiguous blocks—monotone or swapping—that can be merged into hierarchical blocks. The employed method is an instance of the well-known shift-reduce parsing algorithm, and relies on a stack (S) of foreign substrings that have already been translated. Each time the decoder adds a new block to the current translation hypothesis, it shifts the source-language indices of the block onto S, then repeatedly tries reducing the top two elements of S if they are contiguous.[4] This parsing algorithm was first applied in computational geometry to identify convex hulls (Graham, 1972), and its running time was shown to be linear in the length of the sequence (a proof is presented in (Huang et al., 2008), which applies the same algorithm to the binarization of SCFG rules).

Figure 3 provides an example of the execution of this algorithm for the translation output shown in Figure 4, which was produced by a decoder incorporating our hierarchical reordering model. The decoder successively pushes source-language spans [1], [2], [3], which are successively merged into [1-3], and all correspond to monotone orientations.

[3]We would like to point out an inconsistency in Moses between training and testing. Despite the fact that Moses estimates a word-based orientation model during training (i.e., it analyzes the orientation of a given phrase with respect to adjacent word alignments), this model is then treated as a phrase-based orientation model during testing (i.e., as a model that orients phrases with respect to other phrases).

[4]It is not needed to store target-language indices onto the stack, since the decoder proceeds left to right, and thus successive blocks are always contiguous with respect to the target language.

Target phrase	Source	Op.	o_i	Stack
the russian side	[1]	S	M	
hopes	[2]	R	M	[1]
to	[3]	R	M	[1-2]
hold	[11]	S	D	[1-3]
consultations	[12]	R	M	[11], [1-3]
with iran	[9-10]	R	S	[11-12], [1-3]
on this	[6-7]	S	D	[9-12], [1-3]
issue	[8]	R,R	M	[6-7], [9-12], [1-3]
in the near future	[4-5]	R,R	S	[6-12], [1-3]
.	[13]	R,A	M	[1-12]

Figure 3: The application of the shift-reduce parsing algorithm for identifying hierarchical blocks. This execution corresponds to the decoding example of Figure 4. Operations (Op.) include shift (S), reduce (R), and accept (A). The source and stack columns contain source-language spans, which is the only information needed to determine whether two given blocks are contiguous. o_i is the label predicted by the hierarchical model by comparing the current block to the hierarchical phrase that is at the top of the stack.

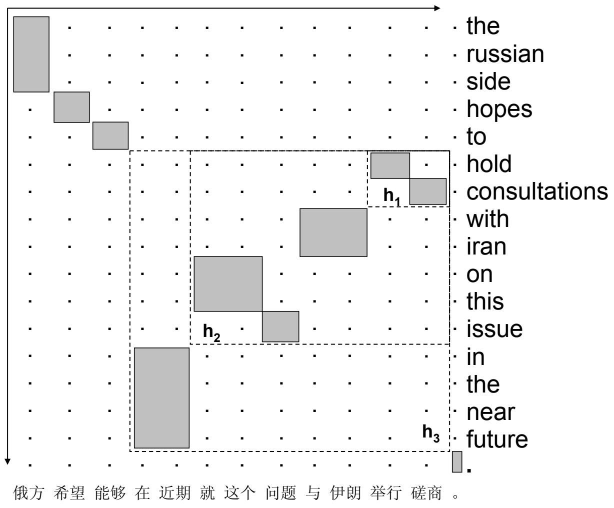


Figure 4: Output of our phrase-based decoder using the hierarchical model on a sentence of MT06. Hierarchical phrases h_1 and h_2 indicate that *with Iran* and *in the near future* have a swap orientation. h_3 indicates that "*to*" and "." are monotone. In this particular example, distortion limit was set to 10.

It then encounters a discontinuity that prevents the next block [11] from being merged with [1-3]. As the decoder reaches the last words of the sentence (*in the near future*), [4-5] is successively merged with [6-12], then [1-3], yielding a stack that contains only [1-12].

A nice property of this parsing algorithm is that it does not worsen the asymptotic running time of beam-search decoders such as Moses (Koehn, 2004a). Such decoders run in time $O(n^2)$, where n is the length of the input sentence. Indeed, each time a partial translation hypothesis is expanded into a longer one, the decoder must perform an $O(n)$ operation in order to copy the coverage set (indicating which foreign words have already been translated) into the new hypothesis. Since this copy operation must be executed $O(n)$ times, the overall time complexity is quadratic. The incorporation of the shift-reduce parser into such a decoder does not worsen overall time complexity: whenever the decoder expands a given partial translation into a longer hypothesis, it simply copies its stack into the newly created hypothesis (similarly to copying the coverage vector, this is an $O(n)$ operation). Hence, the incorporation of the hierarchical models described in the paper into a phrase-based decoder preserves the $O(n^2)$ running time. In practice, we observe based on a set of experiments for Chinese-English and Arabic-English translation that our phrase-based decoder is on average only 1.35 times slower when it is running using hierarchical reordering features and the shift-reduce parser.

We finally note that the decoding algorithm presented in this section can only be applied left-to-right if the decoder itself is operating left-to-right. In order to predict orientations relative to the right-to-left hierarchical reordering model, we must resort to approximations at decoding time. We experimented with different approximations, and the one that worked best (in the experiments discussed in Section 6) is described as follows. First, we note that an analysis of the alignment grid often reveals that certain orientations are impossible. For instance, the block *issue* in Figure 4 can only have discontinuous orientation with respect to what comes next in English, since words surrounding the Chinese phrase have already been translated. When several hierarchical orientations are possible according to the alignment grid, we choose according to the following order of preference: (1) monotone, (2) swap, (3) discontinuous. For instance, in the case of *with iran* in Figure 4, only swap and discontinuous orientations are possible (monotone orientation is impossible because of the block *hold consultations*), hence we give preference to swap. This prediction turns out to be the correct one according to the decoding steps that complete the alignment grid.

5 Discussion

We now analyze the system output of Figure 4 to further motivate the hierarchical model, this time from the perspective of the decoder. We first observe that the prepositional phrase *in the future* should rotate around a relatively large noun phrase headed by *consultations*. Unfortunately, localized reordering models such as (Tillman, 2004) have no means of identifying that such a displacement is a swap (S). According to these models, the orientation of *in the future* with respect to what comes previously is discontinuous (D), which is an uninformative fall-back category. By identifying h_2 (*hold ... issue*) as a hierarchical block, the hierarchical model can properly determine that the block *in the near future* should have a swap orientation.[5] Similar observations can be made regarding blocks h_1 and h_3, which leads our model to predict either monotone orientation (between h_3 and "*to*" and between h_3 and ".") or swap orientation (between h_1 and *with Iran*) while local models would predict discontinuous in all cases.

Another benefit of the hierarchical model is that its representation of phrases remains the same during both training and decoding, which is not the case for word-based and phrase-based reordering models. The deficiency of these local models lies in the fact that blocks handled by phrase-based SMT systems tend to be long at training time and short at test time, which has adverse consequences on non-hierarchical reordering models. For instance, in Figure 4, the phrase-based reordering model categorizes the block *in the near future* as discontinuous, though if the sentence pair had been a training example, this block would count as a swap because of the extracted phrase *on this issue*.

6 Results

In our experiments, we use a re-implementation of the Moses decoder (Koehn et al., 2007). Except for lexical reordering models, all other features are standard features implemented almost exactly as in Moses: four translation features (phrase-based translation probabilities and lexically-weighted probabilities), word penalty, phrase penalty, linear distortion, and language model score. We experiment with two language pairs: Chinese-to-English (C-E) and Arabic-to-English (A-E). For C-E, we trained translation models using a subset of the Chinese-English parallel data released by LDC (mostly news, in particular FBIS and Xinhua News). This subset comprises 12.2M English words, and 11M Chinese words. Chinese words are segmented with a conditional random field (CRF) classifier that conforms to the Chinese Treebank (CTB) standard. The training set for our A-E systems also includes mostly news parallel data released by LDC, and contains 19.5M English words, and 18.7M Arabic tokens that have been segmented using the Arabic Treebank (ATB) (Maamouri et al., 2004) standard.[6]

For our language model, we trained a 5-gram model using the Xinhua and AFP sections of the Gigaword corpus (LDC2007T40), in addition to the target side of the parallel data. For both C-E and A-E, we manually removed documents of Gigaword that were released during periods that overlap with those of our development and test sets. The language model was smoothed with the modified Kneser-Ney algorithm, and we kept only trigrams, 4-grams, and 5-grams that respectively occurred two, three, and three times in the training data.

Parameters were tuned with minimum error-rate training (Och, 2003) on the NIST evaluation set of 2006 (MT06) for both C-E and A-E. Since MERT is prone to search errors, especially with large numbers of parameters, we ran each tuning experiment four times with different initial conditions. This precaution turned out to be particularly important in the case of the combined lexicalized reordering models (the combination of phrase-based and hierarchical discussed later), since MERT must optimize up to 26 parameters at once in these cases.[7] For testing,

[5]Note that the hierarchical phrase *hold ... issue* is not a well-formed syntactic phrase – i.e., it neither matches the bracketing of the verb phrase *hold ... future* nor matches the noun phrase *consultations ... issue* – yet it enables sensible reordering.

[6]Catalog numbers for C-E: LDC2002E18, LDC2003E07, LDC2003E14, LDC2005E83, LDC2005T06, LDC2006E26, and LDC2006E8. For A-E: LDC2007E103, LDC2005E83, LDC2006E24, LDC2006E34, LDC2006E85, LDC2006E92, LDC2007E06, LDC2007E101, LDC2007E46, LDC2007E86, and LDC2008E40.

[7]We combine lexicalized reordering models by simply treating them as distinct features, which incidentally increases the number of model parameters that must be tuned with MERT.

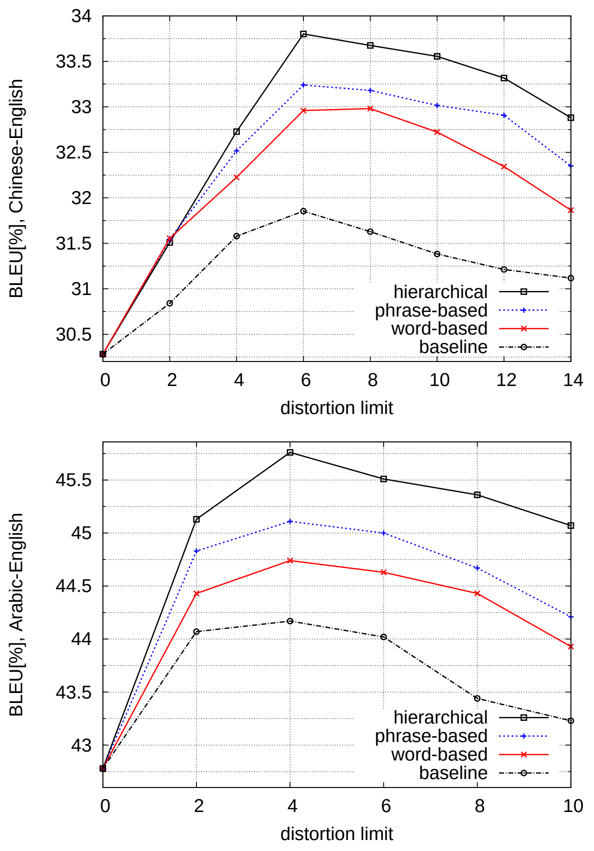


Figure 5: Performance on the Chinese-English and Arabic-English development sets (MT06) with increasing distortion limits for all lexicalized reordering models discussed in the paper. Our novel hierarchical model systematically outperforms all other models for distortion limit equal to or greater than 4. The baseline is Moses with no lexicalized reordering model.

we used the NIST evaluation sets of 2005 and 2008 (MT05 and MT08) for Chinese-English, and the test set of 2005 (MT05) for Arabic-English.

Statistical significance is computed using the approximate randomization test (Noreen, 1989), whose application to MT evaluation (Riezler and Maxwell, 2005) was shown to be less sensitive to type-I errors (i.e., incorrectly concluding that improvement is significant) than the perhaps more widely used bootstrap resampling method (Koehn, 2004b).

Tuning set performance is shown in Figure 5. Since this paper studies various ordering models, it is interesting to first investigate how the distortion limit affects performance.[8] As has been shown in previous work in Chinese-English and Arabic-English translation, limiting phrase displacements to six source-language words is a reasonable choice. For both C-E and A-E, the hierarchical model is significantly better ($p \leq .05$) than either other models for distortion limits equal to or greater than 6 (except for distortion limit 12 in the case of C-E). Since a distortion limit of 6 works reasonably well for both language pairs and is the default in Moses, we used this distortion limit value for all test-set experiments presented in this paper.

LEXICALIZED REORDERING	MT06	MT05	MT08
none	31.85	29.75	25.22
word-based	32.96	31.45	25.86
phrase-based	33.24	31.23	26.01
hierarchical	33.80**	32.20**	26.38
phrase-based + hierarchical	33.86**	32.85**	26.53*

Table 3: BLEU[%] scores (uncased) for Chinese-English and the orientation categories $\{M,S,D\}$. Maximum distortion is set to 6 words, which is the default in Moses. The stars at the bottom of the tables indicate when a given hierarchical model is significantly better than all local models for a given development or test set (*: significance at the .05 level; **: significance at the .01 level).

LEXICALIZED REORDERING	MT06	MT05	MT08
phrase-based	33.79	32.32	26.32
hierarchical	34.01	32.35	26.58
phrase-based + hierarchical	34.36**	32.33	27.03**

Table 4: BLEU[%] scores (uncased) for Chinese-English and the orientation categories $\{M,S,D_l,D_r\}$. Since the distinction between these four categories is not available in Moses, hence we have no baseline results for this case. Maximum distortion is set to 6 words.

Our main results for Chinese-English are shown in Table 3. It appears that hierarchical models provide significant gains over all non-hierarchical models. Improvements on MT06 and MT05 are very significant ($p \leq .01$). In the case of MT08, significant improvement is reached through the combination of both phrase-based and hierarchical models. We often observe substantial gains when we combine such models, presumably because we get the benefit of identifying both local and long-distance swaps.

Since most orientations in the phrase-based model are discontinuous, it is reasonable to ask whether

[8]Note that we ran MERT separately for each distinct distortion limit.

LEXICALIZED REORDERING	MT06	MT05
none	44.03	54.87
word-based	44.64	54.96
phrase-based	45.01	55.09
hierarchical	45.51*	55.50*
phrase-based + hierarchical	45.64**	56.01**

Table 5: BLEU[%] scores (uncased) for Arabic-English and the reordering categories $\{M,S,D\}$.

LEXICALIZED REORDERING	MT06	MT05
phrase-based	44.74	55.52
hierarchical	45.53**	56.02**
phrase-based + hierarchical	45.63**	56.07**

Table 6: BLEU[%] scores (uncased) for Arabic-English and the reordering categories $\{M,S,D_l,D_r\}$.

the relatively poor performance of the phrase-based model is the consequence of an inadequate set of orientation labels. To try to answer this question, we use the set of orientation labels $\{M,S,D_l,D_r\}$ described in Section 3. Results for this different set of orientations are shown in Table 4. While the phrase-based model appears to benefit more from the distinction between left- and right-discontinuous, systems that incorporate hierarchical models remain the most competitive overall: their best performance on MT06, MT05, and MT08 are respectively 34.36, 32.85, and 27.03. The best non-hierarchical models achieve only 33.79, 32.32, and 26.32, respectively. All these differences (i.e., .57, .53, and .71) are statistically significant at the .05 level.

Our results for Arabic-English are shown in Tables 5 and 6. Similarly to C-E, we provide results for two orientation sets: $\{M,S,D\}$ and $\{M,S,D_l,D_r\}$. We note that the four-class orientation set is overall less effective for A-E than for C-E. This is probably due to the fact that there is less probability mass in A-E assigned to the D category, and thus it is less helpful to split the discontinuous category into two.

For both orientation sets, we observe in A-E that the hierarchical model significantly outperforms the local ordering models. Gains provided by the hierarchical model are no less significant than for Chinese-to-English. This positive finding is perhaps a bit surprising, since Arabic-to-English translation generally does not require many word order changes compared to Chinese-to-English translation, and this translation task so far has seldom benefited from hierarchical approaches to MT. In our case, one possible explanation is that Arabic-English translation is benefiting from the fact that orientation predictions of the hierarchical model are consistent across training and testing, which is not the case for the other ordering models discussed in this paper (see Section 4). Overall, hierarchical models are the most effective on the two sets: their best performances on MT06 and MT05 are respectively 45.64 and 56.07. The best non-hierarchical models obtain only 45.01 and 55.52 respectively for the same sets. All these differences (i.e., .63 and .55) are statistically significant at the .05 level.

7 Conclusions and Future Work

In this paper, we presented a lexicalized orientation model that enables phrase movements that are more complex than swaps between adjacent phrases. This model relies on a hierarchical structure that is built as a by-product of left-to-right phrase-based decoding without increase of asymptotic running time. We show that this model provides statistically significant improvements for five NIST evaluation sets and for two language pairs. In future work, we plan to extend the parameterization of our models to not only predict phrase orientation, but also the length of each displacement as in (Al-Onaizan and Papineni, 2006). We believe such an extension would improve translation quality in the case of larger distortion limits. We also plan to experiment with discriminative approaches to estimating reordering probabilities (Zens and Ney, 2006; Xiong et al., 2006), which could also be applied to our work. We think the ability to condition reorderings on any arbitrary feature functions is also very effective in the case of our hierarchical model, since information encoded in the trees would seem beneficial to the orientation prediction task.

8 Acknowledgements

The authors wish to thank the anonymous reviewers for their comments on an earlier draft of this paper. This paper is based on work funded by the Defense Advanced Research Projects Agency through IBM. The content does not necessarily reflect the views of the U.S. Government, and no official endorsement should be inferred.

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 the 44th annual meeting of the ACL (COLING/ACL)*, pages 529–536, Morristown, NJ, USA.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 263–270, June.

Ronald L. Graham. 1972. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1(4):132–133.

Liang Huang, Hao Zhang, Daniel Gildea, and Kevin Knight. 2008. Binarization of synchronous context-free grammars. Technical report, University of Pennsylvania.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*.

Philipp Koehn, Hieu Hoang, Alexandra Birch Mayne, Christopher Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL), Demonstration Session*.

Philipp Koehn. 2004a. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proceedings of the Sixth Conference of the Association for Machine Translation in the Americas*, pages 115–124.

Philipp Koehn. 2004b. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 388–395.

M. Maamouri, A. Bies, T. Buckwalter, and W. Mekki. 2004. The Penn Arabic treebank: Building a large-scale annotated Arabic corpus.

I. Dan Melamed. 2003. Multitext grammars and synchronous parsers. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 79–86.

Eric W. Noreen. 1989. *Computer Intensive Methods for Testing Hypotheses. An Introduction*. Wiley, New York.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

F. Och, D. Gildea, S. Khudanpur, A. Sarkar, K. Yamada, A. Fraser, S. Kumar, L. Shen, D. Smith, K. Eng, V. Jain, Z. Jin, and D. Radev. 2004. A smorgasbord of features for statistical machine translation. In *Proceedings of HLT-NAACL*.

Franz Josef Och. 2003. Minimum error rate training for statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Stefan Riezler and John T. Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 57–64, June.

Christoph Tillman. 2004. A unigram orientation model for statistical machine translation. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 101–104.

Taro Watanabe, Hajime Tsukada, and Hideki Isozaki. 2006. Left-to-right target generation for hierarchical phrase-based translation. In *ACL '06: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 777–784.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 521–528.

Richard Zens and Herman Ney. 2006. Discriminative reordering models for statistical machine translation. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL): Proceedings of the Workshop on Statistical Machine Translation*, pages 55–63, New York City, NY, June.

Language and Translation Model Adaptation using Comparable Corpora

Matthew Snover and **Bonnie Dorr**
Laboratory for Computational Linguistics
and Information Processing
Institute for Advanced Computer Studies
Department of Computer Science
University of Maryland
College Park, MD 20742
{snover, bonnie}@umiacs.umd.edu

Richard Schwartz
BBN Technologies
10 Moulton Street
Cambridge, MA 02138, USA
schwartz@bbn.com

Abstract

Traditionally, statistical machine translation systems have relied on parallel bi-lingual data to train a translation model. While bi-lingual parallel data are expensive to generate, monolingual data are relatively common. Yet monolingual data have been under-utilized, having been used primarily for training a language model in the target language. This paper describes a novel method for utilizing monolingual target data to improve the performance of a statistical machine translation system on news stories. The method exploits the existence of comparable text—multiple texts in the target language that discuss the same or similar stories as found in the source language document. For every source document that is to be translated, a large monolingual data set in the target language is searched for documents that might be comparable to the source documents. These documents are then used to adapt the MT system to increase the probability of generating texts that resemble the comparable document. Experimental results obtained by adapting both the language and translation models show substantial gains over the baseline system.

1 Introduction

While the amount of parallel data available to train a statistical machine translation system is sharply limited, vast amounts of monolingual data are generally available, especially when translating to languages such as English. Yet monolingual data are generally only used to train the language model of the translation system. Previous work (Fung and Yee, 1998; Rapp, 1999) has sought to learn new translations for words by looking at comparable, but not parallel, corpora in multiple languages and analyzing the co-occurrence of words, resulting in the generation of new word-to-word translations.

More recently, Resnik and Smith (2003) and Munteanu and Marcu (2005) have exploited monolingual data in both the source and target languages to find document or sentence pairs that appear to be parallel. This newly discovered bilingual data can then be used as additional training data for the translation system. Such methods generally have a very low yield leaving vast amounts of data that is only used for language modeling.

These methods rely upon comparable corpora, that is, multiple corpora that are of the same general genre. In addition to this, documents can be comparable—two documents that are both on the same event or topic. Comparable documents occur because of the repetition of information across languages, and in the case of news data, on the fact that stories reported in one language are often reported in another language. In cases where no direct translation can be found for a source document, it is often possible to find documents in the target language that are on the same story, or even on a related story, either in subject matter or historically. Such documents can be classified as comparable to the original source document. Phrases within this comparable document are likely to be translations of phrases in the source document, even if the documents themselves are not parallel.

Figure 1 shows an excerpt of the reference translation of an Arabic document, and figure 2 shows a

Cameras are flashing and reporters are following up, for Hollywood star Angelina Jolie is finally talking to the public after a one-month stay in India, but not as a movie star. The Hollywood actress, goodwill ambassador of the United Nations high commissioner for refugees, met with the Indian minister of state for external affairs, Anand Sharma, here today, Sunday, to discuss issues of refugees and children. **...** Jolie, accompanied by her five-year-old son, Maddox, visited the refugee camps that are run by the Khalsa Diwan Society for social services and the high commissioner for refugees Saturday afternoon after she arrived in Delhi. Jolie has been in India since October 5th shooting the movie "A Mighty Heart," which is based on the life of Wall Street Journal correspondent Daniel Pearl, who was kidnapped and killed in Pakistan. Jolie plays the role of Pearl's wife, Mariane.

Figure 1: Excerpt of Example Reference Translation of an Arabic Source Document

Actress Angelina Jolie hopped onto a crowded Mumbai commuter train Monday to film a scene for a movie about slain journalist Daniel Pearl, who lived and worked in India's financial and entertainment capital. Hollywood actor Dan Futterman portrays Pearl and Jolie plays his wife Mariane in the "A Mighty Heart" co-produced by Plan B, a production company founded by Brad Pitt and his ex-wife, actress Jennifer Aniston. Jolie and Pitt, accompanied by their three children – Maddox, 5, 18-month-old Zahara and 5-month-old Shiloh Nouvel – arrived in Mumbai on Saturday from the western Indian city Pune where they were shooting the movie for nearly a month. **...**

Figure 2: Excerpt of Example Comparable Document

comparable passage.[1] In this case, the two new stories are not translations of each other and were not reported at the same time—the comparable passage being an older news story—but both discuss actress Angelina Jolie's visit to India. Many phrases and words are shared between the two, including: the name of the movie, the name and relationship of the actress' character, the name and age of her son and many others. Such a pairing is extremely comparable, although even less related document pairs could easily be considered comparable.

We seek to take advantage of these comparable documents to inform the translation of the source document. This can be done by augmenting the major components of the statistical translation system: the Language Model and the Translation Model. This work is in the same tradition as Kim and Khudanpur (2003), Zhao et al. (2004), and Kim (2005). Kim (2005) used large amounts of comparable data to adapt language models on a document-by-document basis, while Zhao et al. (2004) used comparable data to perform sentence level adaptation of the language model. These adapted language models were shown to improve performance for both automatic speech recognition as well as machine translation.

In addition to language model adaptation we also modify the translation model, adding additional translation rules that enable the translation of new words and phrases in both the source and target languages, as well as increasing the probability of existing translation rules. Translation adaptation using the translation system's own output, known as Self-Training (Ueffing, 2006) has previously shown gains by augmenting the translation model with additional translation rules. In that approach however, the translation model was augmented using parallel data, rather than comparable data, by interpolating a translation model trained using the system output with the original translation model.

Translation model adaptation using comparable out-of-domain parallel data, rather than monolingual data was shown by Hildebrand et al. (2005) to yield significant gains over a baseline system. The translation model was adapted by selecting comparable sentences from parallel corpora for each of the sentences to be translated. In addition to selecting out-of-domain data to adapt the translation model, comparable data selection techniques have been used to select and weight portions of the existing training data for the translation model to improve translation performance (Lu et al., 2007).

The research presented in this paper utilizes a different approach to translation model adaptation using comparable monolingual text rather than parallel text, exploiting data that would otherwise be unused

[1]This is an actual source document from the tuning set used in our experiments, and the first of a number of similar passages found by the comparable text selection system described in section 2.

for estimating the translation model. In addition, this data also informs the translation system by interpolating the original language model with a new language model trained from the same comparable documents.

We discuss the selection of comparable text for model adaptation in section 2. In sections 3.1 and 3.2, we describe the model adaptation for the language model and translation model, respectively. Experimental results describing the application of model adaptation to a hierarchical Arabic-to-English MT system are presented in section 4. Finally we draw conclusions in sections 5.

2 Comparable Text Selection

Comparable text is selected for every source document from a large monolingual corpus in the target language. In practice, one could search the World Wide Web for documents that are comparable to a set of source documents, but this approach presents problems for ensuring the quality of the retrieved documents. The experiments in this paper use comparable text selected from a collection of English news texts. Because these texts are all fluent English, and of comparable genre to the test set, they are also used for training the standard language model training.

The problem of selecting comparable text has been widely studied in the information retrieval community and cross-lingual information retrieval (CLIR) (Oard and Dorr, 1998; Levow et al., 2005) has been largely successful at the task of selecting comparable or relevant documents in one language given a query in another language. We use CLIR to select a ranked list of documents in our target language, English in the experiments described in this paper, for each source document, designated as the query in the CLIR framework, that we wish to translate.

The CLIR problem can be framed probabilistically as: Given a query Q, find a document D that maximizes the equation $\Pr(D \text{ is rel}|Q)$. This equation can be expanded using Bayes' Law as shown in equation 1. The prior probability of a document being relevant can be viewed as uniform, and thus in this work, we assume $\Pr(D \text{ is rel})$ is a constant.[2]

[2]In fact, it can be beneficial to use features of the document to estimate $\Pr(D \text{ is rel})$ (Miller and Schwartz, 1998) but we have not explored that here.

The $\Pr(Q)$ is constant across all documents. Therefore finding a document to maximize $\Pr(D \text{ is rel}|Q)$ is equivalent to finding a document that maximizes $\Pr(Q|D \text{ is rel})$.

$$\Pr(D \text{ is rel}|Q) = \frac{\Pr(D \text{ is rel}) \Pr(Q|D \text{ is rel})}{\Pr(Q)} \tag{1}$$

A method of calculating the probability of a query given a document was proposed by (Xu et al., 2001)[3] and is shown in Equation 2. In this formulation, each foreign word, f, in the query is generated from the foreign vocabulary with probability α and from the English document with probability $1 - \alpha$, where α is a constant.[4] The probability of f being generated by the general foreign vocabulary, F, is $\Pr(f|F) = \mathit{freq}(f, F)/|F|$, the frequency of the word f in the vocabulary divided by the size of the vocabulary. The probability of the word being generated by the English document is the sum of the probabilities of it being generated by each English word, e, in the document which is the frequency of the English word in the document, ($\Pr(e|D) = \mathit{freq}(e, D)/|D|$) multiplied by the probability of the translation of the English word to the foreign word, $\Pr(f|e)$.

$$\Pr(Q|D) = \prod_{f \in Q} (\alpha \Pr(f|F) + (1 - \alpha) \sum_e \Pr(e|D) \Pr(f|e)) \tag{2}$$

This formulation favors longer English documents over shorter English documents. In addition, many documents cover multiple stories and topics. For the purposes of adaptation, shorter, fully comparable documents are preferred to longer, only partially comparable documents. We modify the CLIR system by taking the 1000 highest ranked target language documents found by the CLIR system for each source document, and dividing them into overlapping passages of approximately 300 words.[5] Sen-

[3]Xu et al. (2001) formulated this for the selection of foreign documents given an English query. We reverse this to select English documents given a foreign query.

[4]As in Xu et al. (2001), a value of 0.3 was used for α.

[5]The length of 300 was chosen as this was approximately the same length as the source documents.

tence boundaries are preserved when creating passages, insuring that the text is fluent within each passage. These passages are then scored again by the CLIR system, resulting in a list of passages of about 300 words each for each source document. Finally, we select the top N passages to be used for adaptation.

The N passages selected by this method are not guaranteed to be comparable and are often largely unrelated to the story or topic in the source document. We shall refer to the set of passages selected by the CLIR system as the *bias text* to differentiate it from comparable text, as the adaptation methods will use this text to *bias* the MT system so that its output will be more similar to the bias text.

While we have not conducted experiments using other CLIR systems, the adaptation methods presented in this paper could be applied without modification using another CLIR system, as the adaptation method treats the CLIR system as a black box. With the exception of running a second pass of CLIR, we use the algorithm of Xu et al. (2001) without any significant modification, including the use of a stop word list for both the English and foreign texts. The parameters for $\Pr(f|F)$ and $\Pr(f|e)$ were estimated using the same parallel data that our translation system was trained on.

The bias text selected for a source document is used to adapt the language model (described in section 3.1) and the translation model (described in section 3.2) when translating that source document.

3 Model Adaptation

We use the same bias text to adapt both the language model and the translation model. For language model adaptation, we increase the probability of the word sequences in the bias text, and for translation model adaptation we use additional phrasal translation rules. The adaptations can be done independently and while they can augment each other when used together, this is not required. It is not necessary to use the same number of passages for both forms of adaptation, although doing so makes it more likely both that the English side of the new translation rule will be assigned a high probability by the adapted language model, and that the translation model produces the English text to which the language model has been adapted. Bias text that is used by one adaptation but not the other will receive no special treatment by the other model. This could result in new translation rules that produce text to which the language assigns low probability, or it could result in the language model being able to assign a high probability to a good English translation that cannot be produced by the translation model due to a lack of necessary translation rules.

While both adaptation methods are integrated into a hierarchical translation model (Chiang, 2005), they are largely implementation independent. Language model adaptation could be integrated into any statistical machine translation that uses a language model over words, while translation model adaptation could be added to any statistical machine translation that can utilize phrasal translation rules.

3.1 Language Model Adaptation

For every source document, we estimate a new language model, the *bias language model*, from the corresponding bias text. Since this bias text is short, the corresponding bias language model is small and specific, giving high probabilities to those phrases that occur in the bias text. The bias language model is interpolated with the *generic language model* that would otherwise be used for translation if no LM adaptation was used. The new bias language model is of the same order as the generic language model, so that if a trigram language model is used for the MT decoding, then the biased language model will also be a trigram language model. The bias language model is created using the same settings as the generic language model. In our particular implementation however, the generic language model uses Kneser-Ney smoothing, while the biased language model uses Witten-Bell smoothing due to implementation limitations. In principle the biased language model can be smoothed in the same manner as the generic language model.

We interpolate the bias language model and the generic language model as shown in equation 3, where $\Pr_g$ and $\Pr_b$ are the probabilities from the generic language model and the bias language model, respectively. A constant interpolation weight, λ is used to weight the two probabilities for all documents. While a value for λ could be chosen that minimizes perplexity on a tuning set, in a similar fashion to Kim (2005), it is unclear that such a weight would be ideal when the interpolated language model is used as part of a statistical translation system. In practice we have observed that weights other than one that minimizes perplexity, typically a lower weight, can yield better translation results on the tuning set.

$$\Pr(e) = (1 - \lambda) \Pr_g(e) + \lambda \Pr_b(e) \tag{3}$$

The resulting interpolated language model is then used in place of the generic language model in the translation process, increasing the probability that the translation output will resemble the bias text. It is important to note that, unlike the translation model adaptation described in section 3.2, no new information is added to the system with language model adaptation. Because the bias text is extracted from the same monolingual corpus that the generic language model was estimated from, all of the word sequences used for training the bias language model were also used for training the generic language model. Language model adaptation only increases the weight of the portion of the language model data that was selected as comparable.

3.2 Translation Model Adaptation

It is frequently the case in machine translation that unknown words or phrases are present in the source document, or that the known translations of source words are based on a very small number of occurrences in the training data. In other cases, translations may be known for individual words in the source document, but not for longer phrases. Translation model adaptation seeks to generate new phrasal translation rules for these source words and phrases. The bias text for a source document may, if comparable, contain a number of English words and phrases that are the English side of these desired rules.

Because the source data and the bias text are not translations of each other and are not sentence aligned, conventional alignment tools, such as GIZA++ (Och and Ney, 2000), cannot be used to align the source and bias text. Because the passages in the bias text are not translations of the source document, it will always be the case that portions of the source document have no translation in the bias text, and portions of the bias text have no translation in the source document. In addition a phrase in one of these texts might have multiple, differing translations in the other text.

Unlike language model adaptation, the entirety of the bias text is not used for translation adaptation. We extract those phrases that occur in at least M of the passages in the bias texts. A phrase is only counted once for every passage in which it occurs, so that repeated use of a phrase within a passage does not affect whether it used to generate new rules. Typically, passages selected by the CLIR tend to be very similar to each other if they are comparable to the source document and are very different from each other if they are not comparable to the source document. Phrases that are identical across passages are the ones that are most likely to be comparable, whereas a phrase or word that occurs in only one passage is likely to be present only by chance or if the passage it is in is not comparable. Filtering the target phrases to those that occur in multiple passages therefore serves not only to reduce the total number of rules, but also to filter out phrases from passages that are not comparable.

For each phrase in the source document we generate a new translation to each of the phrases selected from the bias text, and assign it a low uniform probability.[6] For each translation rule we also have a lexical translation probability that we estimate correctly from the trained word model. These new rules are then added to the phrase table of the existing translation model when translating the source document. Rather than adding probability to the existing generic rules, the new rules are marked as bias rules by the system and given their own feature weight. While the vast majority of these rules are incorrect translations, these incorrect rules will be naturally biased against by the translation system. If the source side of a translation already has a number of observed translations, then the low probability of the new bias rule will cause it to not be selected by the translation system. If the new translation rules would produce garbled English, then it will be biased against by the language model. When this is combined with the language model adapta-

[6]A probability of 1/700 is arbitrarily used for the bias rules although it is then weighted by the bias translation rule weight.

tion, a natural pressure is exerted to use the bias rules for source phrases primarily when it would cause the output to look more like the bias text.

4 Experimental Results

We evaluated the performance of language and translation model adaptation with our translation system on two conditions, the details of which are presented in section 4.1. One condition involved a small amount of parallel training, such as one might find when translating a less commonly taught language (LCTL). The other condition involved the full amount of training available for Arabic-to-English translation. In the case of LCTLs we expect our translation model to have the most deficiencies and be most in need of additional translation rules. So, it is under such a condition we would expect the translation model adaptation to be the most beneficial. We evaluate the system's performance under this condition in section 4.2. The effectiveness of this technique on state-of-the-art systems, and its efficiency when used with a well trained generic translation model is presented in section 4.3.

4.1 Implementation Details

Both language-model and translation-model adaptation are implemented on top of a hierarchical Arabic-to-English translation system with string-to-dependency rules as described in Shen et al. (2008). While generalized rules are generated from the parallel data, rules generated by the translation model adaptation are not generalized and are used only as phrasal rules. A trigram language model was used during decoding, and a 5-gram language model was used to re-score the n-best list after decoding. In addition to the features described in Shen et al. (2008), a new feature is added to the model for the bias rule weight, allowing the translation system to effectively tune the probability of the rules added by translation model adaptation in order to improve performance on the tuning set.

Bias texts were selected from three monolingual corpora: the English Gigaword corpus (2,793,350,201 words), the FBIS corpus (28,465,936 words), and a collection of news archive data collected from the websites of various online, public news sites (828,435,409 words). All three corpora were also part of the generic language model training data. Language model adaptation on both the trigram and 5-gram language models used 10 comparable passages with an interpolation weight of 0.1. Translation model adaptation used 10 comparable passages for the bias text and a value of 2 for M.

Each selected passage contains approximately 300 words, so in the case where 10 comparable passages are used to create a bias text, the resulting text will be 3000 words long on average. The language models created using these bias texts are very specific giving large probability to n-gram sequences seen in those texts.

The construction of the bias texts increases the overall run-time of the translation system, although in practice this is a small expenditure. The most intensive portion is the initial indexing of the monolingual corpus, but this is only required once and can be reused for any subsequent test set that is evaluated. This index can then be quickly searched for comparable passages. When considering research environments, test sets are used repeatedly and bias texts only need to be built once per set, making the building cost negligible. Otherwise, the time required to build the bias text is still small compared to the actual translation time.

All conditions were optimized using BLEU (Papineni et al., 2002) and evaluated using both BLEU and Translation Edit Rate (TER) (Snover et al., 2006). BLEU is an accuracy measure, so higher values indicate better performance, while TER is an error metric, so lower values indicate better performance. Optimization was performed on a tuning set of newswire data, comprised of portions of MTEval 2004, MTEval 2005, and GALE 2007 newswire development data, a total of 48921 words of English in 1385 segments and 173 documents. Results were measured on the NIST MTEval 2006 Arabic Evaluation set, which was 55578 words of English in 1797 segments and 104 documents. Four reference translations were used for scoring each translation.

Parameter optimization method was done using n-best optimization, although the adaptation process is not tied to this method. The MT decoder is run on the tuning set generating an n-best list (where $n = 300$), on which all of the translation features (including bias rule weights) are optimized using

Powell's method. These new weights are then used to decode again, repeating the whole process, using a cumulative n-best list. This continues for several iterations until performance on the tuning set stabilizes. The resulting feature weights are used when decoding the test set. A similar, but simpler, method is used to determine the feature weights after 5-gram rescoring. This n-best optimization method has subtle implications for translation model adaptation. In the first iteration, few bias rules are used in decoding the 300-best, and those that are used frequently help, although the overall gain is small due to the small number of bias rules used. This causes the optimizer to greatly increase the weight of the bias rules, causing the decoder to overuse the bias rules in the next iteration causing a sharp decrease in translation quality. Several iterations are needed for the cumulative n-best to achieve sufficient diversity and size to assign a weight for the bias translation rules that results in an increase in performance over the baseline. Alternative optimization methods could likely circumvent this process. Language model adaptation does not suffer from this phenomenon.

4.2 Less Commonly Taught Language Simulation

In order to better examine the nature of translation model adaptation, we elected to work with a translation model that was trained on only 5 million words of parallel Arabic-English text. Limiting the translation model training in this way simulates the problem of translating less commonly taught languages (LCTL) where less parallel text is available, a situation that is not the case for Arabic. Since the model is trained on less parallel data, it is lacking a large number of translation rules, which is expected to be addressed by the translation model adaptation. By working in an environment with a more deprived baseline translation model, we are giving the translation model adaptation more room to assist.

The experiments described below use a 5 million word Arabic parallel text corpus constructed from the LDC2004T18 and LDC2006E25 corpora. The full monolingual English data were used for the language model and for selection of comparable documents. Unless otherwise specified no language model adaptation was used.

We first establish an upper limit on the gain using translation model adaptation, using the reference data to adapt the translation system. These reference data can be considered to be extremely comparable, better than could ever be hoped to gain by comparable document selection. We first aligned this data using GIZA++ to the source data, simulating the ideal case where we can perfectly determine which source words translate to which comparable words. Because our translation model adaptation system assigns uniform probability to all bias rules, we ignore the correct rule probabilities that we could extract from word alignment and assign uniform probability to all of the bias translation rules. As expected, this gives a large gain over the baseline.

We also examine limiting these new translation rules to those rules whose target side occurs in the top 100 passages selected by CLIR, thus minimizing the adaption to those rules that it theoretically could learn from the bias text. On average, 50% of the rules were removed by this filtering, resulting in a corresponding 50% decrease in the gain over the baseline. The results of these experiments and an unadapted baseline are shown in table 1.

Test Set	TM Adaptation	TER	BLEU
Tune	None	0.4984	0.4080
	Aligned Reference	0.3692	0.5841
	Overlapping Only	0.4179	0.5138
MT06	None	0.5516	0.3468
	Aligned Reference	0.4517	0.5216
	Overlapping Only	0.4899	0.4335

Table 1: LCTL Aligned Reference Adaptation Results

The fair translation model adaptation system, however, does not align source phrases to the correct bias text phrases in such a fashion, and instead aligns all source words to all target words. To investigate the effect of this over production of rules, we again used the reference translations as if they were comparable data, but we ignored the alignments learned by GIZA++, and instead allowed all source phrases to translate to all English phrases in the reference text, with uniform probability. This still shows large gains in translation quality over the baseline, as measured by TER and BLEU. Again, we also examined limiting the text used for translation model adaptation to those phrases that occur in both the reference text and the top 100 comparable passages selected the CLIR system. While this decreased performance, the system still performs significantly better than the baseline, as shown in the following table 2.

Test Set	TM Adaptation	TER	BLEU
Tune	None	0.4984	0.4080
	Unaligned Ref.	0.4492	0.4566
	Overlapping Only	0.4808	0.4313
MT06	None	0.5516	0.3468
	Unaligned Ref.	0.5254	0.3990
	Overlapping Only	0.5390	0.3695

Table 2: LCTL Unaligned Reference Adaptation Results

Applying translation model and language model adaptation fairly, using only bias text from the comparable data selection, yields smaller gains on both the tuning and MT06 sets, as shown in table 3. The combination of language-model and translation-model adaptation exceeds the gains that would be achieved over the baseline by either method separately.

Test Set	Adaptation	TER	BLEU
Tune	None	0.4984	0.4080
	LM	0.4922	0.4140
	TM	0.4916	0.4169
	LM & TM	0.4888	0.4244
MT06	None	0.5516	0.3468
	LM	0.5559	0.3490
	TM	0.5545	0.3478
	LM & TM	0.5509	0.3536

Table 3: LCTL Fair Adaptation Results

4.3 Full Parallel Training Results

While the simulation described in section 4.2 used only 5 million words of parallel training, 230 million words of parallel data from 18.5 million segments were used for training the full Arabic-to-English translation system. This parallel data includes the LDC2007T08 ”ISI Arabic-English Automatically Extracted Parallel Text” corpus (Munteanu and Marcu, 2007), which was created from monolingual corpora in English and Arabic using the algorithm described in Munteanu and Marcu (2005), as the techniques used in that work are separate and independent from the adaptation methods we describe in this paper.[7] Language model adaptation and translation model adaptation were applied both independently and jointly to the translation system, and the results were evaluated against an unadapted baseline, as shown in table 4.

While gains from language model adaptation were substantial on the tuning set, on the MT06 test set they are reduced to a 0.65% gain on BLEU and a negligible improvement in TER. The translation model adaptation performs better with 1.37% improvement in BLEU and a 0.26% improvement in TER. This gain increases to a 2.07% improvement in BLEU and a 0.64% improvement in TER when language adaptation is used in conjunction with the translation model adaptation, showing the importance of using both adaptation methods. While it could be expected that a more heavily trained translation model might not require the benefit of language and translation model adaptation, a more substantial gain over the baseline can be seen when both forms of adaptation are used than in the case with less parallel training—a difference of 2.07% BLEU versus 0.68% BLEU.

Test Set	Adaptation	TER	BLEU
Tune	None	0.4339	0.4661
	LM	0.4227	0.4857
	TM	0.4351	0.4657
	LM & TM	0.4245	0.4882
MT06	None	0.5146	0.3852
	LM	0.5140	0.3917
	TM	0.5120	0.3989
	LM & TM	0.5082	0.4059

Table 4: Full Training Adaptation Results

Of the comparable passages selected by the CLIR system for the MT06 test set in the full training experiment, 16.3% were selected from the News

[7]The two methods are not directly comparable, and so we do not make any attempt to do so. Munteanu and Marcu (2005) creates new parallel corpora from two monolingual corpora. This new parallel data is generally applicable for training a translation model but does not target any particular test set. Our adaptation method does not generate new parallel data, but creates a new specific translation model for a test document that is being translated.

Archive corpus, 81.2% were selected from the English GigaWord corpus and 2.5% were selected from the FBIS corpus. A slightly different distribution was found for the Tuning set, where 17.8% of the passages were selected from the News Archive corpus, 77.1% were selected from the English GigaWord corpus, and 5.1% were selected from the FBIS corpus.

5 Discussion

The reuse of a monolingual corpus that was already used by a translation system for language model training to perform both language and translation model adaptation shows large gains over an unadapted baseline. By leveraging off of a CLIR system, which itself contains no information not already given to the translation system,[8] potentially comparable passages can be found which allow improved translation. Surprisingly, these gains are largest when the baseline model is better trained, indicating that a strong reliance of the adaptation on the existing models.

One explantation for these counter-intuitive results–larger gains in the full training scenario versus the LCTL scenario–is that the lexical probabilities are better estimated in the former case. The bias rules all have equal translation probability and only vary in probability according to the lexical probability of the rules. Better estimates of these lexical probabilities may enable the translation system to more clearly distinguish between helpful and harmful bias rules.

There are many clear directions for the improvement of these methods. The current adaptation method does not utilize the probabilities from the CLIR system and treats the top-ranked passages all as equally comparable regardless of the probability assigned. Variable weighting of passages could prove beneficial to both language model adaptation, where the passages could be weighted proportionally to the probability of the passage being relevant, and translation model adaptation, where the requirement on repetition of phrases across passages could be weighted, as could the probability of the new rules produced by the translation system. In addition, the CLIR score, among other possible features such as phrase overlap, could be used to determine those documents where no comparable passage could be detected and where it would be beneficial to not adapt the models.

[8]The probabilistic parameters of the CLIR system are estimated from the same parallel corpora that is used to train the generic translation model.

A clear limitation of using comparable documents to adapt the language and translation model is that comparable documents must be found. For many source documents, none of the top passages found by the CLIR system were comparable. We suspect that while this will always occur to some extent, this becomes more common as the monolingual data becomes less like the source data, such as when there is a large time gap between the two. The full extent of this and the effect of the level of document comparability on translation remains an open question. In addition, while newswire is an excellent source of comparable text, it is unclear how well this method can be used on newsgroups or spoken data, where the fluency of the source text is diminished. When translating news stories, this technique is not limited to major news events. While many of the events discussed in the source data receive world-wide attention, many are local events that are unreported in the English comparable data used in our experiments. Events of a similar nature or events involving many of the same people often do occur in the English comparable data, allowing improvement even when the stories are quite different.

The adaptation methods described in this paper are not limited to a particular framework of statistical machine translation, but have applicability to any statistical machine translation system that uses a language model or translation rules.

Acknowledgments

This work was supported, in part, by BBN Technologies under the GALE Program, DARPA/IPTO Contract No. HR0011-06-C-0022. 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 sponsor. We are very grateful to all three reviewers for their careful and thoughtful reviews.

References

David Chiang. 2005. A Hierarchical Phrase-Based Model for Statistical Machine Translation. In *Proceedings of ACL*, pages 263–270.

Pascale Fung and Lo Yuen Yee. 1998. An IR Approach for Translating New Words from Nonparallel, Comparable Texts. In *Proceedings of COLING-ACL98*, pages 414–420, August.

Almut Silja Hildebrand, Matthias Eck, Stephan Vogel, and Alex Waibel. 2005. Adaptation of the Translation Model for Statistical Machine Translation based on Information Retrieval. In *Proceedings of EAMT 2005*, Budapest, Hungary, May.

Woosung Kim and Sanjeev Khudanpur. 2003. Cross-Lingual Lexical Triggers in Statistical Language Modeling. In *2003 Conference on Empirical Methods in Natural Language Processing (EMNLP 2003)*, pages 17–24, July.

Woosung Kim. 2005. *Language Model Adaptation for Automatic Speech Recognition and Statistical Machine Translation*. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD.

Gina-Anne Levow, Douglas W. Oard, and Philip Resnik. 2005. Dictionary-based cross-language retrieval. *Information Processing and Management*, 41:523–547.

Yajuan Lu, Jin Huang, and Qun Liu. 2007. Improving Statistical Machine Translation Performance by Training Data Selection and Optimization. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 343–350.

T. Leek Miller and Richard Schwartz. 1998. BBN at TREC7: Using Hidden Markov Models for Information Retrieval. In *TREC 1998*, pages 80–89, Gaithersburg, MD.

Dragos Stefan Munteanu and Daniel Marcu. 2005. Improving Machine Translation Performance by Exploiting Non-Parallel Corpora. *Computational Linguistics*, 31:477–504.

Dragos Stefan Munteanu and Daniel Marcu. 2007. Isi arabic-english automatically extracted parallel text. Linguistic Data Consortium, Philadelphia.

Douglas W. Oard and Bonnie J. Dorr. 1998. Evaluating Cross-Language Text Retrieval Effectiveness. In Gregory Grefenstette, editor, *Cross-Language Information Retrieval*, pages 151–161. Kluwer Academic Publishers, Boston, MA.

F. J. Och and H. Ney. 2000. Improved Statistical Alignment Models. In *Proceedings of the 38th Annual Conference of the Association for Computational Linguistics*, pages 440–447, Hongkong, China.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Traslation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated english and german corpora. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 519–526.

Philip Resnik and Noah Smith. 2003. The Web as a Parallel Corpus. *Computational Linguistics*, 29:349–380.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A New String-to-Dependency Machine Translation Algorithm with a Target Dependency Language Model. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, June.

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 Association for Machine Translation in the Americas*.

Nicola Ueffing. 2006. Using Monolingual Source-Language to Improve MT Performance. In *Proceedings of IWSLT 2006*.

Jinxi Xu, Ralpha Weischedel, and Chanh Nguyen. 2001. Evaluating a Probabilistic Model for Cross-lingual Information Retrieval. In *Proceedings of SIGIR 2001 Conference*, pages 105–110.

Bing Zhao, Matthias Eck, and Stephan Vogel. 2004. Language Model Adaptation for Statistical Machine Translation via Structured Query Models. In *Proceedings of Coling 2004*, pages 411–417, Geneva, Switzerland, Aug 23–Aug 27. COLING.

Sparse Multi-Scale Grammars for Discriminative Latent Variable 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 a discriminative, latent variable approach to syntactic parsing in which rules exist at multiple scales of refinement. The model is formally a latent variable CRF grammar over trees, learned by iteratively splitting grammar productions (not categories). Different regions of the grammar are refined to different degrees, yielding grammars which are three orders of magnitude smaller than the single-scale baseline and 20 times smaller than the split-and-merge grammars of Petrov et al. (2006). In addition, our discriminative approach integrally admits features beyond local tree configurations. We present a multi-scale training method along with an efficient CKY-style dynamic program. On a variety of domains and languages, this method produces the best published parsing accuracies with the smallest reported grammars.

1 Introduction

In latent variable approaches to parsing (Matsuzaki et al., 2005; Petrov et al., 2006), one models an observed treebank of coarse *parse* trees using a grammar over more refined, but unobserved, *derivation* trees. The parse trees represent the desired output of the system, while the derivation trees represent the typically much more complex underlying syntactic processes. In recent years, latent variable methods have been shown to produce grammars which are as good as, or even better than, earlier parsing work (Collins, 1999; Charniak, 2000). In particular, in Petrov et al. (2006) we exhibited a very accurate category-splitting approach, in which a coarse initial grammar is refined by iteratively splitting each grammar category into two subcategories using the EM algorithm. Of course, each time the number of grammar categories is doubled, the number of binary productions is increased by a factor of eight. As a result, while our final grammars used few categories, the number of total active (non-zero) productions was still substantial (see Section 7). In addition, it is reasonable to assume that some generatively learned splits have little discriminative utility. In this paper, we present a discriminative approach which addresses both of these limitations.

We introduce *multi-scale* grammars, in which some productions reference fine categories, while others reference coarse categories (see Figure 2). We use the general framework of *hidden variable CRFs* (Lafferty et al., 2001; Koo and Collins, 2005), where gradient-based optimization maximizes the likelihood of the observed variables, here parse trees, summing over log-linearly scored derivations. With multi-scale grammars, it is natural to refine *productions* rather than categories. As a result, a category such as NP can be complex in some regions of the grammar while remaining simpler in other regions. Additionally, we exploit the flexibility of the discriminative framework both to improve the treatment of unknown words as well as to include *span features* (Taskar et al., 2004), giving the benefit of some input features integrally in our dynamic program. Our multi-scale grammars are 3 orders of magnitude smaller than the fully-split baseline grammar and 20 times smaller than the generative split-and-merge grammars of Petrov et al. (2006).

In addition, we exhibit the best parsing numbers on several metrics, for several domains and languages.

Discriminative parsing has been investigated before, such as in Johnson (2001), Clark and Curran (2004), Henderson (2004), Koo and Collins (2005), Turian et al. (2007), Finkel et al. (2008), and, most similarly, in Petrov and Klein (2008). However, in all of these cases, the final parsing performance fell short of the best generative models by several percentage points or only short sentences were used. Only in combination with a generative model was a discriminative component able to produce high parsing accuracies (Charniak and Johnson, 2005; Huang, 2008). Multi-scale grammars, in contrast, give higher accuracies using smaller grammars than previous work in this direction, outperforming top generative models in grammar size and in parsing accuracy.

2 Latent Variable Parsing

Treebanks are typically not annotated with fully detailed syntactic structure. Rather, they present only a coarse trace of the true underlying processes. As a result, learning a grammar for parsing requires the estimation of a more highly articulated model than the naive CFG embodied by such treebanks. A manual approach might take the category NP and subdivide it into one subcategory NPˆS for subjects and another subcategory NPˆVP for objects (Johnson, 1998; Klein and Manning, 2003). However, rather than devising linguistically motivated features or splits, latent variable parsing takes a fully automated approach, in which each symbol is split into unconstrained subcategories.

2.1 Latent Variable Grammars

Latent variable grammars augment the treebank trees with latent variables at each node. This creates a set of (exponentially many) *derivations* over split categories for each of the original *parse trees* over unsplit categories. For each observed category A we now have a set of latent subcategories A_x. For example, NP might be split into NP$_1$ through NP$_8$.

The parameters of the refined productions $A_x \rightarrow B_y\ C_z$, where A_x is a subcategory of A, B_y of B, and C_z of C, can then be estimated in various ways; past work has included both generative (Matsuzaki et al., 2005; Liang et al., 2007) and discriminative approaches (Petrov and Klein, 2008). We take the discriminative log-linear approach here. Note that the comparison is only between estimation methods, as Smith and Johnson (2007) show that the model classes are the same.

2.2 Log-Linear Latent Variable Grammars

In a log-linear latent variable grammar, each production $r = A_x \rightarrow B_y\ C_z$ is associated with a multiplicative weight ϕ_r (Johnson, 2001; Petrov and Klein, 2008) (sometimes we will use the log-weight θ_r when convenient). The probability of a derivation t of a sentence w is proportional to the product of the weights of its productions r:

$$P(t|w) \propto \prod_{r \in t} \phi_r$$

The score of a parse T is then the sum of the scores of its derivations:

$$P(T|w) = \sum_{t \in T} P(t|w)$$

3 Hierarchical Refinement

Grammar refinement becomes challenging when the number of subcategories is large. If each category is split into k subcategories, each (binary) production will be split into k^3. The resulting memory limitations alone can prevent the practical learning of highly split grammars (Matsuzaki et al., 2005). This issue was partially addressed in Petrov et al. (2006), where categories were repeatedly split and some splits were re-merged if the gains were too small. However, while the grammars are indeed compact at the (sub-)category level, they are still dense at the production level, which we address here.

As in Petrov et al. (2006), we arrange our subcategories into a hierarchy, as shown in Figure 1. In practice, the construction of the hierarchy is tightly coupled to a split-based learning process (see Section 5). We use the naming convention that an original category A becomes A_0 and A_1 in the first round; A_0 then becoming A_{00} and A_{01} in the second round, and so on. We will use $\hat{x} \succ x$ to indicate that the subscript or subcategory x is a refinement of $\hat{x}$.[1] We

[1] Conversely, $\hat{x}$ is a coarser version of x, or, in the language of Petrov and Klein (2007), $\hat{x}$ is a projection of x.

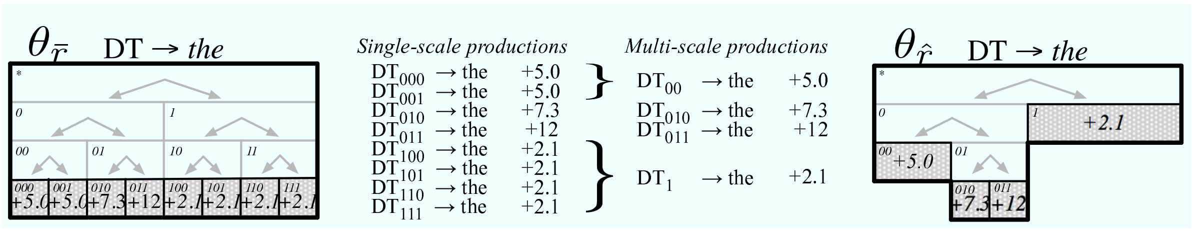


Figure 1: Multi-scale refinement of the $DT \rightarrow the$ production. The multi-scale grammar can be encoded much more compactly than the equally expressive single scale grammar by using only the shaded features along the fringe.

will also say that $\hat{x}$ dominates x, and $\overline{x}$ will refer to fully refined subcategories. The same terminology can be applied to (binary) productions, which split into eight refinements each time the subcategories are split in two.

The core observation leading to multi-scale grammars is that when we look at the refinements of a production, many are very similar in weight. It is therefore advantageous to record productions only at the level where they are distinct from their children in the hierarchy.

4 Multi-Scale Grammars

A multi-scale grammar is a grammar in which some productions reference fine categories, while others reference coarse categories. As an example, consider the multi-scale grammar in Figure 2, where the NP category has been split into two subcategories (NP_0, NP_1) to capture subject and object distinctions. Since *it* can occur in subject and object position, the production NP $\rightarrow$ *it* has remained unsplit. In contrast, in a single-scale grammar, two productions $\mathrm{NP}_0 \rightarrow$ *it* and $\mathrm{NP}_1 \rightarrow$ *it* would have been necessary. We use * as a wildcard, indicating that NP_* can combine with any other NP, while NP_1 can only combine with other NP_1. Whenever subcategories of different granularity are combined, the resulting constituent takes the more specific label.

In terms of its structure, a multi-scale grammar is a set of productions over varyingly refined symbols, where each production is associated with a weight. Consider the refinement of the production shown in Figure 1. The original unsplit production (at top) would naively be split into a tree of many subproductions (downward in the diagram) as the grammar categories are incrementally split. However, it may be that many of the fully refined productions share the same weights. This will be especially common in the present work, where we go out of our way to achieve it (see Section 5). For example, in Figure 1, the productions $\mathrm{DT}_x \rightarrow$ *the* have the same weight for all categories DT_x which refine DT_1.[2] A multi-scale grammar can capture this behavior with just 4 productions, while the single-scale grammar has 8 productions. For binary productions the savings will of course be much higher.

In terms of its semantics, a multi-scale grammar is simply a compact encoding of a fully refined latent variable grammar, in which identically weighted refinements of productions have been collapsed to the coarsest possible scale. Therefore, rather than attempting to control the degree to which categories are split, multi-scale grammars simply encode productions at varying scales. It is hence natural to speak of refining productions, while considering the categories to exist at all degrees of refinement. Multi-scale grammars enable the use of coarse (even unsplit) categories in some regions of the grammar, while requiring very specific subcategories in others, as needed. As we will see in the following, this flexibility results in a tremendous reduction of grammar parameters, as well as improved parsing time, because the vast majority of productions end up only partially split.

Since a multi-scale grammar has productions which can refer to different levels of the category hierarchy, there must be constraints on their coherence. Specifically, for each fully refined production, exactly one of its dominating coarse productions must be in the grammar. More formally, the multi-scale grammar partitions the space of fully refined base rules such that each $\overline{r}$ maps to a unique

[2]We define dominating productions and refining productions analogously as for subcategories.

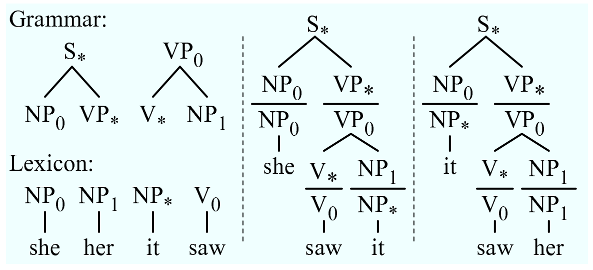


Figure 2: In multi-scale grammars, the categories exist at varying degrees of refinement. The grammar in this example enforces the correct usage of *she* and *her*, while allowing the use of *it* in both subject and object position.

dominating rule $\hat{r}$, and for all base rules $\overline{r}'$ such that $\hat{r} \succ \overline{r}'$, $\overline{r}'$ maps to $\hat{r}$ as well. This constraint is always satisfied if the multi-scale grammar consists of fringes of the production refinement hierarchies, indicated by the shading in Figure 1.

A multi-scale grammar straightforwardly assigns scores to derivations in the corresponding fully refined single scale grammar: simply map each refined derivation rule to its dominating abstraction in the multi-scale grammar and give it the corresponding weight. The fully refined grammar is therefore trivially (though not compactly) reconstructable from its multi-scale encoding.

It is possible to directly define a derivational semantics for multi-scale grammars which does not appeal to the underlying single scale grammar. However, in the present work, we use our multi-scale grammars only to compute expectations of the underlying grammars in an efficient, implicit way.

5 Learning Sparse Multi-Scale Grammars

We now consider how to discriminatively learn multi-scale grammars by iterative splitting productions. There are two main concerns. First, because multi-scale grammars are most effective when many productions share the same weight, sparsity is very desirable. In the present work, we exploit L_1-regularization, though other techniques such as structural zeros (Mohri and Roark, 2006) could also potentially be used. Second, training requires repeated parsing, so we use coarse-to-fine chart caching to greatly accelerate each iteration.

5.1 Hierarchical Training

We learn discriminative multi-scale grammars in an iterative fashion (see Figure 1). As in Petrov et al. (2006), we start with a simple X-bar grammar from an input treebank. The parameters θ of the grammar (production log-weights for now) are estimated in a log-linear framework by maximizing the penalized log conditional likelihood $\mathcal{L}_{cond} - R(\theta)$, where:

$$\begin{aligned} \mathcal{L}_{cond}(\theta) &= \log \prod_i \mathrm{P}(T_i|w_i) \\ R(\theta) &= \sum_r |\theta_r| \end{aligned}$$

We directly optimize this non-convex objective function using a numerical gradient based method (LBFGS (Nocedal and Wright, 1999) in our implementation). To handle the non-diferentiability of the L_1-regularization term $R(\theta)$ we use the orthant-wise method of Andrew and Gao (2007). Fitting the log-linear model involves the following derivatives:

$$\frac{\partial \mathcal{L}_{cond}(\theta)}{\partial \theta_r} = \sum_i \Big(\mathbb{E}_\theta \left[f_r(t) | T_i \right] - \mathbb{E}_\theta [f_r(t) | w_i] \Big)$$

where the first term is the expected count f_r of a production r in derivations corresponding to the correct parse tree T_i and the second term is the expected count of the production in all derivations of the sentence w_i. Note that r may be of any scale. As we will show below, these expectations can be computed exactly using marginals from the chart of the inside/outside algorithm (Lari and Young, 1990).

Once the base grammar has been estimated, all categories are split in two, meaning that all binary productions are split in eight. When splitting an already refined grammar, we only split productions whose log-weight in the previous grammar deviates from zero.[3] This creates a refinement hierarchy over productions. Each newly split production r is given a unique feature, as well as inheriting the features of its parent productions $\hat{r} \succ r$:

$$\phi_r = \exp \Big(\sum_{\hat{r} \succ r} \theta_{\hat{r}} \Big)$$

The parent productions $\hat{r}$ are then removed from the grammar and the new features are fit as described

[3] L_1-regularization drives more than 95% of the feature weights to zero in each round.

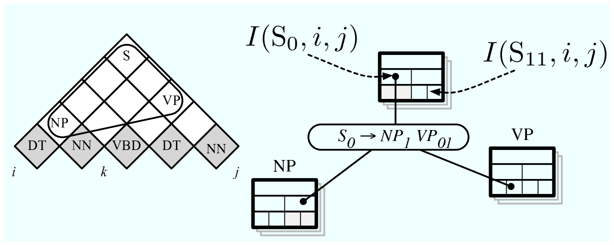


Figure 3: A multi-scale chart can be used to efficiently compute inside/outside scores using productions of varying specificity.

above. We detect that we have split a production too far when all child production features are driven to zero under L_1 regularization. In such cases, the children are collapsed to their parent production, which forms an entry in the multi-scale grammar.

5.2 Efficient Multi-Scale Inference

In order to compute the expected counts needed for training, we need to parse the training set, score all derivations and compute posteriors for all subcategories in the refinement hierarchy. The inside/outside algorithm (Lari and Young, 1990) is an efficient dynamic program for summing over derivations under a context-free grammar. It is fairly straightforward to adapt this algorithm to multi-scale grammars, allowing us to sum over an exponential number of derivations *without* explicitly reconstructing the underlying fully split grammar.

For single-scale latent variable grammars, the inside score $I(A_{\overline{x}}, i, j)$ of a fully refined category $A_{\overline{x}}$ spanning $\langle i, j\rangle$ is computed by summing over all possible productions $\overline{r} = A_{\overline{x}} \rightarrow B_{\overline{y}}\ C_{\overline{z}}$ with weight $\phi_{\overline{r}}$, spanning $\langle i, k\rangle$ and $\langle k, j\rangle$ respectively:[4]

$$I(A_{\overline{x}}, i, j) = \sum_{\overline{r}} \phi_{\overline{r}} \sum_{k} I(B_{\overline{y}}, i, k) I(C_{\overline{z}}, k, j)$$

Note that this involves summing over *all* relevant fully refined grammar productions.

The key quantities we will need are marginals of the form $I(A_x, i, j)$, the sum of the scores of all fully refined derivations rooted at any $A_{\overline{x}}$ dominated by A_x and spanning $\langle i, j\rangle$. We define these marginals in terms of the standard inside scores of the most refined subcategories $A_{\overline{x}}$:

$$I(A_x, i, j) = \sum_{\overline{x} \prec x} I(A_{\overline{x}}, i, j)$$

When working with multi-scale grammars, we expand the standard three-dimensional chart over spans and grammar categories to store the scores of all subcategories of the refinement hierarchy, as illustrated in Figure 3. This allows us to compute the scores more efficiently by summing only over rules $\hat{r} = A_{\hat{x}} \rightarrow B_{\hat{y}}\ C_{\hat{z}} \succ \overline{r}$:

$$\begin{aligned}
I(A_{\overline{x}}, i, j) &= \sum_{\hat{r}} \sum_{\overline{r} \prec \hat{r}} \phi_{\overline{r}} \sum_{k} I(B_{\overline{y}}, i, k) I(C_{\overline{z}}, k, j) \\
&= \sum_{\hat{r}} \phi_{\hat{r}} \sum_{\overline{r} \prec \hat{r}} \sum_{k} I(B_{\overline{y}}, i, k) I(C_{\overline{z}}, k, j) \\
&= \sum_{\hat{r}} \phi_{\hat{r}} \sum_{\overline{y} \prec \hat{y}} \sum_{\overline{z} \prec \hat{z}} \sum_{k} I(B_{\overline{y}}, i, k) I(C_{\overline{z}}, k, j) \\
&= \sum_{\hat{r}} \phi_{\hat{r}} \sum_{k} \sum_{\overline{y} \prec \hat{y}} I(B_{\overline{y}}, i, k) \sum_{\overline{z} \prec \hat{z}} I(C_{\overline{z}}, k, j) \\
&= \sum_{\hat{r}} \phi_{\hat{r}} \sum_{k} I(B_{\hat{y}}, i, k) I(C_{\hat{z}}, k, j)
\end{aligned}$$

Of course, some of the same quantities are computed repeatedly in the above equation and can be cached in order to obtain further efficiency gains. Due to space constraints we omit these details, and also the computation of the outside score, as well as the handling of unary productions.

5.3 Feature Count Approximations

Estimating discriminative grammars is challenging, as it requires repeatedly taking expectations over all parses of all sentences in the training set. To make this computation practical on large data sets, we use the same approach as Petrov and Klein (2008). Therein, the idea of coarse-to-fine parsing (Charniak et al., 1998) is extended to handle the repeated parsing of the same sentences. Rather than computing the entire coarse-to-fine history in every round of training, the pruning history is cached between training iterations, effectively avoiding the repeated calculation of similar quantities and allowing the efficient approximation of feature count expectations.

[4]These scores lack any probabilistic interpretation, but can be normalized to compute the necessary expectations for training (Petrov and Klein, 2008).

6 Additional Features

The discriminative framework gives us a convenient way of incorporating additional, overlapping features. We investigate two types of features: unknown word features (for predicting the part-of-speech tags of unknown or rare words) and span features (for determining constituent boundaries based on individual words and the overall sentence shape).

6.1 Unknown Word Features

Building a parser that can process arbitrary sentences requires the handling of previously unseen words. Typically, a classification of rare words into word classes is used (Collins, 1999). In such an approach, the word classes need to be manually defined *a priori*, for example based on discriminating word shape features (suffixes, prefixes, digits, etc.).

While this component of the parsing system is rarely talked about, its importance should not be underestimated: when using only one unknown word class, final parsing performance drops several percentage points. Some unknown word features are universal (e.g. digits, dashes), but most of them will be highly language dependent (prefixes, suffixes), making additional human expertise necessary for training a parser on a new language. It is therefore beneficial to automatically learn what the discriminating word shape features for a language are. The discriminative framework allows us to do that with ease. In our experiments we extract prefixes and suffixes of length ≤ 3 and add those features to words that occur 25 times or less in the training set. These unknown word features make the latent variable grammar learning process more language independent than in previous work.

6.2 Span Features

There are many features beyond local tree configurations which can enhance parsing discrimination; Charniak and Johnson (2005) presents a varied list. In reranking, one can incorporate any such features, of course, but even in our dynamic programming approach it is possible to include features that decompose along the dynamic program structure, as shown by Taskar et al. (2004). We use non-local *span features*, which condition on properties of input spans (Taskar et al., 2004). We illustrate our span features with the following example and the span $\langle 1, 4 \rangle$:

$$_0 \text{“} \ _1 [\, \textit{Yes} \ _2 \text{”} \ _3 \, , \,] \ _4 \textit{he} \ _5 \textit{said} \ _6 \, . \ _7$$

We first added the following lexical features:

- the first (*Yes*), last (*comma*), preceding (“) and following (*he*) words,
- the word pairs at the left edge $\langle$“,*Yes*$\rangle$, right edge $\langle$*comma,he*$\rangle$, inside border $\langle$*Yes,comma*$\rangle$ and outside border $\langle$“,*he*$\rangle$.

Lexical features were added for each span of length three or more. We used two groups of span features, one for natural constituents and one for synthetic ones.[5] We found this approach to work slightly better than anchoring the span features to particular constituent labels or having only one group.

We also added shape features, projecting the sentence to abstract shapes to capture global sentence structures. Punctuation shape replaces every non-punctuation word with `x` and then further collapses strings of `x` to `x+`. Our example becomes `#‘‘x’’,x+.#`, and the punctuation feature for our span is `‘‘[x’’,]x`. Capitalization shape projects the example sentence to `#.X..xx.#`, and `.[X..]x` for our span. Span features are a rich source of information and our experiments should be seen merely as an initial investigation of their effect in our system.

7 Experiments

We ran experiments on a variety of languages and corpora using the standard training and test splits, as described in Table 1. In each case, we start with a completely unannotated X-bar grammar, obtained from the raw treebank by a simple right-branching binarization scheme. We then train multi-scale grammars of increasing latent complexity as described in Section 5, directly incorporating the additional features from Section 6 into the training procedure. Hierarchical training starting from a raw treebank grammar and proceeding to our most refined grammars took three days in a parallel implementation using 8 CPUs. At testing time we marginalize out the hidden structure and extract the tree with the highest number of expected correct productions, as in Petrov and Klein (2007).

[5]Synthetic constituents are nodes that are introduced during binarization.

	Training Set	Dev. Set	Test Set
ENGLISH-WSJ (Marcus et al., 1993)	Sections 2-21	Section 22	Section 23
ENGLISH-BROWN (Francis et al. 2002)	see ENGLISH-WSJ	10% of the data[6]	10% of the the data[6]
FRENCH[7] (Abeille et al., 2000)	Sentences 1-18,609	Sentences 18,610-19,609	Sentences 19,609-20,610
GERMAN (Skut et al., 1997)	Sentences 1-18,602	Sentences 18,603-19,602	Sentences 19,603-20,602

Table 1: Corpora and standard experimental setups.

We compare to a baseline of discriminatively trained latent variable grammars (Petrov and Klein, 2008). We also compare our discriminative multi-scale grammars to their generative split-and-merge cousins, which have been shown to produce the state-of-the-art figures in terms of accuracy and efficiency on many corpora. For those comparisons we use the grammars from Petrov and Klein (2007).

7.1 Sparsity

One of the main motivations behind multi-scale grammars was to create compact grammars. Figure 4 shows parsing accuracies vs. grammar sizes. Focusing on the grammar size for now, we see that multi-scale grammars are extremely compact - even our most refined grammars have less than 50,000 active productions. This is 20 times smaller than the generative split-and-merge grammars, which use explicit category merging. The graph also shows that this compactness is due to controlling production sparsity, as the single-scale discriminative grammars are two orders of magnitude larger.

7.2 Accuracy

Figure 4 shows development set results for English. In terms of parsing accuracy, multi-scale grammars significantly outperform discriminatively trained single-scale latent variable grammars and perform on par with the generative split-and-merge grammars. The graph also shows that the unknown word and span features each add about 0.5% in final parsing accuracy. Note that the span features improve the performance of the unsplit baseline grammar by 8%, but not surprisingly their contribution gets smaller when the grammars get more refined. Section 8 contains an analysis of some of the learned features, as well as a comparison between discriminatively and generatively trained grammars.

[6]See Gildea (2001) for the exact setup.

[7]This setup contains only sentences without annotation errors, as in (Arun and Keller, 2005).

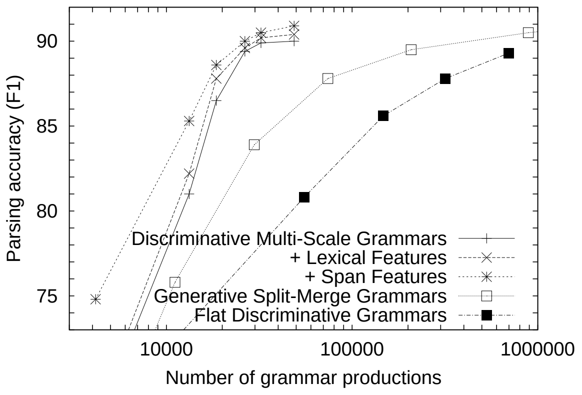


Figure 4: Discriminative multi-scale grammars give similar parsing accuracies as generative split-merge grammars, while using an order of magnitude fewer rules.

7.3 Efficiency

Petrov and Klein (2007) demonstrates how the idea of coarse-to-fine parsing (Charniak et al., 1998; Charniak et al., 2006) can be used in the context of latent variable models. In coarse-to-fine parsing the sentence is rapidly pre-parsed with increasingly refined grammars, pruning away unlikely chart items in each pass. In their work the grammar is projected onto coarser versions, which are then used for pruning. Multi-scale grammars, in contrast, do not require projections. The refinement hierarchy is built in and can be used directly for coarse-to-fine pruning. Each production in the grammar is associated with a set of hierarchical features. To obtain a coarser version of a multi-scale grammar, one therefore simply limits which features in the refinement hierarchy can be accessed. In our experiments, we start by parsing with our coarsest grammar and allow an additional level of refinement at each stage of the pre-parsing. Compared to the generative parser of Petrov and Klein (2007), parsing with multi-scale grammars requires the evaluation of 29% fewer productions, decreasing the average parsing time per sentence by 36% to 0.36 sec/sentence.

	≤ 40 words		all	
Parser	F1	EX	F1	EX
ENGLISH-WSJ				
Petrov and Klein (2008)	88.8	35.7	88.3	33.1
Charniak et al. (2005)	90.3	39.6	89.7	37.2
Petrov and Klein (2007)	**90.6**	39.1	**90.1**	37.1
This work w/o span features	89.7	39.6	89.2	37.2
This work w/ span features	90.0	**40.1**	89.4	**37.7**
ENGLISH-WSJ (reranked)				
Huang (2008)	**92.3**	**46.2**	**91.7**	**43.5**
ENGLISH-BROWN				
Charniak et al. (2005)	84.5	34.8	82.9	31.7
Petrov and Klein (2007)	84.9	34.5	83.7	31.2
This work w/o span features	85.3	35.6	84.3	32.1
This work w/ span features	**85.6**	**35.8**	**84.5**	**32.3**
ENGLISH-BROWN (reranked)				
Charniak et al. (2005)	**86.8**	**39.9**	**85.2**	**37.8**
FRENCH				
Arun and Keller (2005)	79.2	21.2	75.6	16.4
This Paper	**80.1**	**24.2**	**77.2**	**19.2**
GERMAN				
Petrov and Klein (2007)	80.8	40.8	80.1	39.1
This Paper	**81.5**	**45.2**	**80.7**	**43.9**

Table 2: Our final test set parsing accuracies compared to the best previous work on English, French and German.

7.4 Final Results

For each corpus we selected the grammar that gave the best performance on the development set to parse the final test set. Table 2 summarizes our final test set performance, showing that multi-scale grammars achieve state-of-the-art performance on most tasks. On WSJ-English, the discriminative grammars perform on par with the generative grammars of Petrov et al. (2006), falling slightly short in terms of F1, but having a higher exact match score. When trained on WSJ-English but tested on the Brown corpus, the discriminative grammars clearly outperform the generative grammars, suggesting that the highly regularized and extremely compact multi-scale grammars are less prone to overfitting. All those methods fall short of reranking parsers like Charniak and Johnson (2005) and Huang (2008), which, however, have access to many additional features, that cannot be used in our dynamic program.

When trained on the French and German treebanks, our multi-scale grammars achieve the best figures we are aware of, without any language specific modifications. This confirms that latent variable models are well suited for capturing the syntactic properties of a range of languages, and also shows that discriminative grammars are still effective when trained on smaller corpora.

8 Analysis

It can be illuminating to see the subcategories that are being learned by our discriminative multi-scale grammars and to compare them to generatively estimated latent variable grammars. Compared to the generative case, the lexical categories in the discriminative grammars are substantially less refined. For example, in the generative case, the nominal categories were fully refined, while in the discriminative case, fewer nominal clusters were heavily used. One reason for this can be seen by inspecting the first two-way split in the NNP tag. The generative model split into initial NNPs (*San, Wall*) and final NNPs (*Francisco, Street*). In contrast, the discriminative split was between organizational entities (*Stock, Exchange*) and other entity types (*September, New, York*). This constrast is unsurprising. Generative likelihood is advantaged by explaining lexical choice – *New* and *York* occur in very different slots. However, they convey the same information about the syntactic context above their base NP and are therefore treated the same, discriminatively, while the systematic attachment distinctions between temporals and named entities are more predictive.

Analyzing the syntactic and semantic patterns learned by the grammars shows similar trends. In Table 3 we compare the number of subcategories in the generative split-and-merge grammars to the average number of features per unsplit production with that phrasal category as head in our multi-scale grammars after 5 split (and merge) rounds. These quantities are inherently different: the number of features should be roughly cubic in the number of subcategories. However, we observe that the numbers are very close, indicating that, due to the sparsity of our productions, and the efficient multi-scale encoding, the number of grammar parameters grows linearly in the number of subcategories. Furthermore, while most categories have similar complexity in those two cases, the complexity of the two most refined phrasal categories are flipped. Generative grammars split NPs most highly, discrimina-

	NP	VP	PP	S	SBAR	ADJP	ADVP	QP	PRN
Generative subcategories	32	24	20	12	12	12	8	7	5
Discriminative production parameters	19	32	20	14	14	8	7	9	6

Table 3: Complexity of highly split phrasal categories in generative and discriminative grammars. Note that subcategories are compared to production parameters, indicating that the number of parameters grows cubicly in the number of subcategories for generative grammars, while growing linearly for multi-scale grammars.

	ENGLISH	GERMAN	FRENCH
Adjectives	-ous -ble -nth	-los -bar -ig	-ien -ble -ive
Nouns	-ion -en -cle	-tät -ung -rei	-té -eur -ges
Verbs	-ed -s	-st -eht	-ées -é
Adverbs	-ly	-mal	-ent
Numbers	-ty	-zig	—

Table 4: Automatically learned suffixes with the highest weights for different languages and part-of-speech tags.

tive grammars split the VP. This distinction seems to be because the complexity of VPs is more syntactic (e.g. complex subcategorization), while that of NPs is more lexical (noun choice is generally higher entropy than verb choice).

It is also interesting to examine the automatically learned word class features. Table 4 shows the suffixes with the highest weight for a few different categories across the three languages that we experimented with. The learning algorithm has selected discriminative suffixes that are typical derviational or inflectional morphemes in their respective languages. Note that the highest weighted suffixes will typically not correspond to the most common suffix in the word class, but to the most discriminative.

Finally, the span features also exhibit clear patterns. The highest scoring span features encourage the words between the last two punctuation marks to form a constituent (excluding the punctuation marks), for example `,[x+].` and `:[x+].` Words between quotation marks are also encouraged to form constituents: ``` ``[x+]'' ``` and ``` x[``x+'']x ```. Span features can also discourage grouping words into constituents. The features with the highest negative weight involve single commas: `x[x,x+]`, and `x[x+,x+]x` and so on (indeed, such spans were structurally disallowed by the Collins (1999) parser).

9 Conclusions

Discriminatively trained multi-scale grammars give state-of-the-art parsing performance on a variety of languages and corpora. Grammar size is dramatically reduced compared to the baseline, as well as to methods like split-and-merge (Petrov et al., 2006). Because fewer parameters are estimated, multi-scale grammars may also be less prone to overfitting, as suggested by a cross-corpus evaluation experiment. Furthermore, the discriminative framework enables the seamless integration of additional, overlapping features, such as span features and unknown word features. Such features further improve parsing performance and make the latent variable grammars very language independent.

Our parser, along with trained grammars for a variety of languages, is available at http://nlp.cs.berkeley.edu.

References

A. Abeille, L. Clement, and A. Kinyon. 2000. Building a treebank for French. In *2nd International Conference on Language Resources and Evaluation*.

G. Andrew and J. Gao. 2007. Scalable training of L1-regularized log-linear models. In *ICML '07*.

A. Arun and F. Keller. 2005. Lexicalization in crosslinguistic probabilistic parsing: the case of french. In *ACL '05*.

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} *Workshop on Very Large Corpora*.

E. Charniak, M. Johnson, D. McClosky, et al. 2006. Multi-level coarse-to-fine PCFG Parsing. In *HLT-NAACL '06*.

E. Charniak. 2000. A maximum–entropy–inspired parser. In *NAACL '00*.

S. Clark and J. R. Curran. 2004. Parsing the WSJ using CCG and log-linear models. In *ACL '04*.

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

J. Finkel, A. Kleeman, and C. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *ACL '08*.

W. N. Francis and H. Kucera. 2002. Manual of information to accompany a standard corpus of present-day edited american english. In *TR, Brown University*.

D. Gildea. 2001. Corpus variation and parser performance. *EMNLP '01*.

J. Henderson. 2004. Discriminative training of a neural network statistical parser. In *ACL '04*.

L. Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *ACL '08*.

M. Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24:613–632.

M. Johnson. 2001. Joint and conditional estimation of tagging and parsing models. In *ACL '01*.

D. Klein and C. Manning. 2003. Accurate unlexicalized parsing. In *ACL '03*, pages 423–430.

T. Koo and M. Collins. 2005. Hidden-variable models for discriminative reranking. In *EMNLP '05*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *ICML '01*.

K. Lari and S. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*.

P. Liang, S. Petrov, M. I. Jordan, and D. Klein. 2007. The infinite PCFG using hierarchical Dirichlet processes. In *EMNLP '07*.

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*.

M. Mohri and B. Roark. 2006. Probabilistic context-free grammar induction based on structural zeros. In *HLT-NAACL '06*.

J. Nocedal and S. J. Wright. 1999. *Numerical Optimization*. Springer.

S. Petrov and D. Klein. 2007. Improved inference for unlexicalized parsing. In *HLT-NAACL '07*.

S. Petrov and D. Klein. 2008. Discriminative log-linear grammars with latent variables. In *NIPS '08*.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *ACL '06*.

W. Skut, B. Krenn, T. Brants, and H. Uszkoreit. 1997. An annotation scheme for free word order languages. In *Conf. on Applied Natural Language Processing*.

N. A. Smith and M. Johnson. 2007. Weighted and probabilistic context-free grammars are equally expressive. *Computational Lingusitics*.

B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. 2004. Max-margin parsing. In *EMNLP '04*.

J. Turian, B. Wellington, and I. D. Melamed. 2007. Scalable discriminative learning for natural language parsing and translation. In *NIPS '07*.

Two Languages are Better than One (for Syntactic Parsing)

David Burkett and Dan Klein
Computer Science Division
University of California, Berkeley
{dburkett,klein}@cs.berkeley.edu

Abstract

We show that jointly parsing a bitext can substantially improve parse quality on both sides. In a maximum entropy bitext parsing model, we define a distribution over source trees, target trees, and node-to-node alignments between them. Features include monolingual parse scores and various measures of syntactic divergence. Using the translated portion of the Chinese treebank, our model is trained iteratively to maximize the marginal likelihood of training tree pairs, with alignments treated as latent variables. The resulting bitext parser outperforms state-of-the-art monolingual parser baselines by 2.5 F_1 at predicting English side trees and 1.8 F_1 at predicting Chinese side trees (the highest published numbers on these corpora). Moreover, these improved trees yield a 2.4 BLEU increase when used in a downstream MT evaluation.

1 Introduction

Methods for machine translation (MT) have increasingly leveraged not only the formal machinery of syntax (Wu, 1997; Chiang, 2007; Zhang et al., 2008), but also linguistic tree structures of either the source side (Huang et al., 2006; Marton and Resnik, 2008; Quirk et al., 2005), the target side (Yamada and Knight, 2001; Galley et al., 2004; Zollmann et al., 2006; Shen et al., 2008), or both (Och et al., 2003; Aue et al., 2004; Ding and Palmer, 2005). These methods all rely on automatic parsing of one or both sides of input bitexts and are therefore impacted by parser quality. Unfortunately, parsing general bitexts well can be a challenge for newswire-trained treebank parsers for many reasons, including out-of-domain input and tokenization issues.

On the other hand, the presence of translation pairs offers a new source of information: bilingual constraints. For example, Figure 1 shows a case where a state-of-the-art English parser (Petrov and Klein, 2007) has chosen an incorrect structure which is incompatible with the (correctly chosen) output of a comparable Chinese parser. Smith and Smith (2004) previously showed that such bilingual constraints can be leveraged to transfer parse quality from a resource-rich language to a resource-impoverished one. In this paper, we show that bilingual constraints and reinforcement can be leveraged to substantially improve parses on both sides of a bitext, even for two resource-rich languages.

Formally, we present a log-linear model over triples of source trees, target trees, and node-to-node tree alignments between them. We consider a set of core features which capture the scores of monolingual parsers as well as measures of syntactic alignment. Our model conditions on the input sentence pair and so features can and do reference input characteristics such as posterior distributions from a word-level aligner (Liang et al., 2006; DeNero and Klein, 2007).

Our training data is the translated section of the Chinese treebank (Xue et al., 2002; Bies et al., 2007), so at training time correct trees are observed on both the source and target side. Gold tree alignments are not present and so are induced as latent variables using an iterative training procedure. To make the process efficient and modular to existing monolingual parsers, we introduce several approximations: use of k-best lists in candidate generation, an adaptive bound to avoid considering all k^2 combinations, and Viterbi approximations to alignment posteriors.

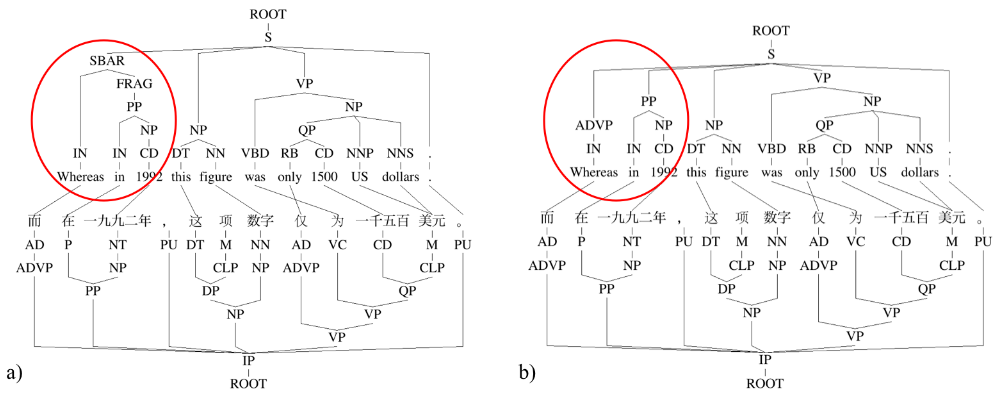

a)
ROOT
S
SBAR
FRAG
PP
NP
VP
NP
QP
IN IN CD DT NN VBD RB CD NNP NNS .
Whereas in 1992 this figure was only 1500 US dollars .
而 在 一九九二年 ， 这 项 数字 仅 为 一千五百 美元 。
AD P NT PU DT M NN AD VC CD M PU
ADVP NP CLP NP ADVP CLP
PP DP QP
NP VP
VP
IP
ROOT
b)
ROOT
S
VP
PP
NP
ADVP NP NP QP
IN IN CD DT NN VBD RB CD NNP NNS .
Whereas in 1992 this figure was only 1500 US dollars .
而 在 一九九二年 ， 这 项 数字 仅 为 一千五百 美元 。
AD P NT PU DT M NN AD VC CD M PU
ADVP NP CLP NP ADVP CLP
PP DP QP
NP VP
VP
IP
ROOT

Figure 1: Two possible parse pairs for a Chinese-English sentence pair. The parses in a) are chosen by independent monolingual statistical parsers, but only the Chinese side is correct. The gold English parse shown in b) is further down in the 100-best list, despite being more consistent with the gold Chinese parse. The circles show where the two parses differ. Note that in b), the ADVP and PP nodes correspond nicely to Chinese tree nodes, whereas the correspondence for nodes in a), particularly the SBAR node, is less clear.

We evaluate our system primarily as a parser and secondarily as a component in a machine translation pipeline. For both English and Chinese, we begin with the state-of-the-art parsers presented in Petrov and Klein (2007) as a baseline. Joint parse selection improves the English trees by 2.5 F_1 and the Chinese trees by 1.8 F_1. While other Chinese treebank parsers do not have access to English side translations, this Chinese figure does outperform all published monolingual Chinese treebank results on an equivalent split of the data.

As MT motivates this work, another valuable evaluation is the effect of joint selection on downstream MT quality. In an experiment using a syntactic MT system, we find that rules extracted from joint parses results in an increase of 2.4 BLEU points over rules extracted from independent parses.[1] In sum, jointly parsing bitexts improves parses substantially, and does so in a way that that carries all the way through the MT pipeline.

2 Model

In our model, we consider pairs of sentences (s, s'), where we use the convention that unprimed variables are source domain and primed variables are target domain. These sentences have parse trees t (respectively t') taken from candidate sets T (T'). Non-terminal nodes in trees will be denoted by n (n') and we abuse notation by equating trees with their node sets. Alignments a are simply at-most-one-to-one matchings between a pair of trees t and t' (see Figure 2a for an example). Note that we will also mention *word* alignments in feature definitions; a and the unqualified term *alignment* will always refer to node alignments. Words in a sentence are denoted by v (v').

[1]It is anticipated that in some applications, such as tree transducer extraction, the alignments themselves may be of value, but in the present work they are not evaluated.

Our model is a general log-linear (maximum entropy) distribution over triples (t, a, t') for sentence pairs (s, s'):

$$\mathrm{P}(t, a, t|s, s') \propto \exp(w^\top \phi(t, a, t'))$$

Features are thus defined over (t, a, t') triples; we discuss specific features below.

3 Features

To use our model, we need features of a triple (t, a, t') which encode both the monolingual quality of the trees as well as the quality of the alignment between them. We introduce a variety of features in the next sections.

3.1 Monolingual Features

To capture basic monolingual parse quality, we begin with a single source and a single target feature whose values are the log likelihood of the source tree t and the target tree t', respectively, as given by our baseline monolingual parsers. These two features are called SOURCELL and TARGETLL respectively. It is certainly possible to augment these simple features with what would amount to monolingual reranking features, but we do not explore that option here. Note that with only these two features, little can be learned: all positive weights w cause the jointly optimal parse pair (t, t') to comprise the two top-1 monolingual outputs (the baseline).

3.2 Word Alignment Features

All other features in our model reference the entire triple (t, a, t'). In this work, such features are defined over aligned node pairs for efficiency, but generalizations are certainly possible.

Bias: The first feature is simply a bias feature which has value 1 on each aligned node pair (n, n'). This bias allows the model to learn a general preference for denser alignments.

Alignment features: Of course, some alignments are better than others. One indicator of a good node-to-node alignment between n and n' is that a good word alignment model thinks that there are many word-to-word alignments in their bispan. Similarly, there should be few alignments that violate that bispan. To compute such features, we define $a(v, v')$ to be the posterior probability assigned to the word alignment between v and v' by an independent word aligner.[2]

Before defining alignment features, we need to define some additional variables. For any node $n \in t$ ($n' \in t'$), the inside span $i(n)$ ($i(n')$) comprises the input tokens of s (s') dominated by that node. Similarly, the complement, the outside span, will be denoted $o(n)$ ($o(n')$), and comprises the tokens not dominated by that node. See Figure 2b,c for examples of the resulting regions.

$$\begin{aligned}
\textsc{InsideBoth} &= \sum_{v \in i(n)} \sum_{v' \in i(n')} a(v, v') \\
\textsc{InSrcOutTrg} &= \sum_{v \in i(n)} \sum_{v' \in o(n')} a(v, v') \\
\textsc{InTrgOutSrc} &= \sum_{v \in o(n)} \sum_{v' \in i(n')} a(v, v')
\end{aligned}$$

[2]It is of course possible to learn good alignments using lexical indicator functions or other direct techniques, but given our very limited training data, it is advantageous to leverage counts from an unsupervised alignment system.

Hard alignment features: We also define the hard versions of these features, which take counts from the word aligner's hard top-1 alignment output δ:

$$\begin{aligned}
\textsc{HardInsideBoth} &= \sum_{v \in i(n)} \sum_{v' \in i(n')} \delta(v, v') \\
\textsc{HardInSrcOutTrg} &= \sum_{v \in i(n)} \sum_{v' \in o(n')} \delta(v, v') \\
\textsc{HardInTrgOutSrc} &= \sum_{v \in o(n)} \sum_{v' \in i(n')} \delta(v, v')
\end{aligned}$$

Scaled alignment features: Finally, undesirable larger bispans can be relatively sparse at the word alignment level, yet still contain many good word alignments simply by virtue of being large. We therefore define a scaled count which measures density rather than totals. The geometric mean of span lengths was a superior measure of bispan "area" than the true area because word-level alignments tend to be broadly one-to-one in our word alignment model.

$$\begin{aligned}
\textsc{ScaledInsideBoth} &= \frac{\textsc{InsideBoth}}{\sqrt{|i(n)| \cdot |i(n')|}} \\
\textsc{ScaledInSrcOutTrg} &= \frac{\textsc{InSrcOutTrg}}{\sqrt{|i(n)| \cdot |o(n')|}} \\
\textsc{ScaledInTrgOutSrc} &= \frac{\textsc{InTrgOutSrc}}{\sqrt{|o(n)| \cdot |i(n')|}}
\end{aligned}$$

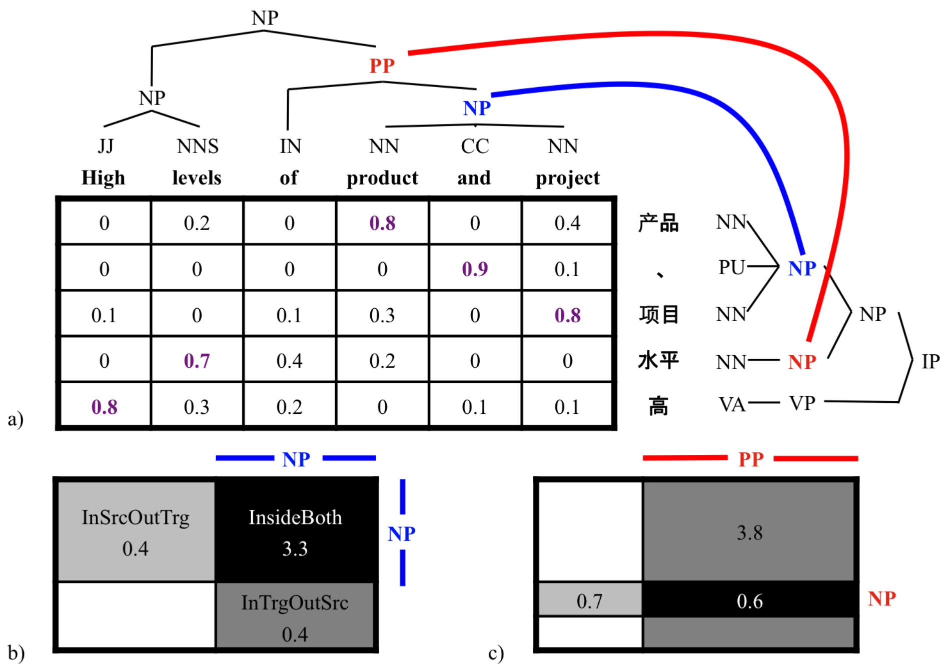
Head word alignment features: When considering a node pair (n, n'), especially one which dominates a large area, the above measures treat all spanned words as equally important. However, lexical heads are generally more representative than other spanned words. Let h select the headword of a node according to standard head percolation rules (Collins, 2003; Bikel and Chiang, 2000).

$$\begin{aligned}
\textsc{AlignHeadWord} &= a(h(n), h(n')) \\
\textsc{HardAlignHeadWord} &= \delta(h(n), h(n'))
\end{aligned}$$

3.3 Tree Structure Features

We also consider features that measure correspondences between the tree structures themselves.

Span difference: We expect that, in general, aligned nodes should dominate spans of roughly the same length, and so we allow the model to learn to



0	0.2	0	**0.8**	0	0.4	产品
0	0	0	0	**0.9**	0.1	、
0.1	0	0.1	0.3	0	**0.8**	项目
0	**0.7**	0.4	0.2	0	0	水平
0.8	0.3	0.2	0	0.1	0.1	高

Figure 2: a) An example of a legal alignment on a Chinese-English sentence fragment with one good and one bad node pair, along with sample word alignment posteriors. Hard word alignments are bolded. b) The word alignment regions for the good NP-NP alignment. InsideBoth regions are shaded in black, InSrcOutTrg in light grey, and InTrgOutSrc in grey. c) The word alignment regions for the bad PP-NP alignment.

penalize node pairs whose inside span lengths differ greatly.

$$\text{SPANDIFF} = ||i(n)| - |i(n')||$$

Number of children: We also expect that there will be correspondences between the rules of the CFGs that generate the trees in each language. To encode some of this information, we compute indicators of the number of children c that the nodes have in t and t'.

$$\text{NUMCHILDREN}\langle|c(n)|, |c(n')|\rangle = 1$$

Child labels: In addition, we also encode whether certain label pairs occur as children of matched nodes. Let $c(n, \ell)$ select the children of n with label ℓ.

$$\text{CHILDLABEL}\langle\ell, \ell'\rangle = |c(n, \ell)| \cdot |c(n', \ell')|$$

Note that the corresponding "self labels" feature is not listed because it arises in the next section as a typed variant of the bias feature.

3.4 Typed vs untyped features

For each feature above (except monolingual features), we create label-specific versions by conjoining the label pair $(\ell(n), \ell(n'))$. We use both the typed and untyped variants of all features.

4 Training

Recall that our data condition supplies sentence pairs (s, s') along with gold parse pairs (g, g'). We do not observe the alignments a which link these parses. In principle, we want to find weights which maximize the marginal log likelihood of what we do observe given our sentence pairs:[3]

$$w^* = \arg\max_w \sum_a \text{P}(g, a, g'|s, s', w) \quad (1)$$

$$= \arg\max_w \frac{\sum_a exp(w^\top \phi(g, a, g'))}{\sum_{(t,t')} \sum_a exp(w^\top \phi(t, a, t'))} \quad (2)$$

There are several challenges. First, the space of symmetric at-most-one-to-one matchings is #P-hard

[3]In this presentation, we only consider a single sentence pair for the sake of clarity, but our true objective was multiplied over all sentence pairs in the training data.

to sum over exactly (Valiant, 1979). Second, even without matchings to worry about, standard methods for maximizing the above formulation would require summation over pairs of trees, and we want to assume a fairly generic interface to independent monolingual parsers (though deeper joint modeling and/or training is of course a potential extension). As we have chosen to operate in a reranking mode over monolingual k-best lists, we have another issue: our k-best outputs on the data which trains our model may not include the gold tree pair. We therefore make several approximations and modifications, which we discuss in turn.

4.1 Viterbi Alignments

Because summing over alignments a is intractable, we cannot evaluate (2) or its derivatives. However, if we restrict the space of possible alignments, then we can make this optimization more feasible. One way to do this is to stipulate in advance that for each tree pair, there is a canonical alignment $a_0(t, t')$. Of course, we want a_0 to reflect actual correspondences between t and t', so we want a reasonable definition that ensures the alignments are of reasonable quality. Fortunately, it turns out that we can efficiently optimize a given a fixed tree pair and weight vector:

$$\begin{aligned} a^* &= \arg\max_a \mathrm{P}(a|t, t', s, s', w) \\ &= \arg\max_a \mathrm{P}(t, a, t'|s, s', w) \\ &= \arg\max_a \exp(w^\top \phi(t, a, t')) \end{aligned}$$

This optimization requires only that we search for an optimal alignment. Because all our features can be factored to individual node pairs, this can be done with the Hungarian algorithm in cubic time.[4] Note that we do not enforce any kind of domination consistency in the matching: for example, the optimal alignment might in principle have the source root aligning to a target non-root and vice versa.

We then define $a_0(t, t')$ as the alignment that maximizes $w_0^\top \phi(t, a, t')$, where w_0 is a fixed initial weight vector with a weight of 1 for InsideBoth, -1 for InSrcOutTrg and InTrgOutSrc, and 0 for all other features. Then, we simplify (2) by fixing the alignments a_0:

$$w^* = \arg\max_w \frac{exp(w^\top \phi(g, a_0(g, g'), g'))}{\sum_{(t,t')} exp(w^\top \phi(t, a_0(t, t'), t'))} \quad (3)$$

This optimization has no latent variables and is therefore convex and straightforward. However, while we did use this as a rapid training procedure during development, fixing the alignments a priori is both unsatisfying and also less effective than a procedure which allows the alignments a to adapt during training.

Again, for fixed alignments a, optimizing w is easy. Similarly, with a fixed w, finding the optimal a for any particular tree pair is also easy. Another option is therefore to use an iterative procedure that alternates between choosing optimal alignments for a fixed w, and then reoptimizing w for those fixed alignments according to (3). By iterating, we perform the following optimization:

$$w^* = \arg\max_w \frac{\max_a exp(w^\top \phi(g, a, g'))}{\sum_{(t,t')} \max_a exp(w^\top \phi(t, a, t'))} \quad (4)$$

Note that (4) is just (2) with summation replaced by maximization. Though we do not know of any guarantees for this EM-like algorithm, in practice it converges after a few iterations given sufficient training data. We initialize the procedure by setting w_0 as defined above.

4.2 Pseudo-gold Trees

When training our model, we approximate the sets of all trees with k-best lists, T and T', produced by monolingual parsers. Since these sets are not guaranteed to contain the gold trees g and g', our next approximation is to define a set of *pseudo-gold trees*, following previous work in monolingual parse reranking (Charniak and Johnson, 2005). We define $\hat{T}$ ($\hat{T}'$) as the F_1-optimal subset of T (T'). We then modify (4) to reflect the fact that we are seeking to maximize the likelihood of trees in this subset:

$$w^* = \arg\max_w \sum_{(t,t') \in (\hat{T}, \hat{T}')} \mathrm{P}(t, t'|s, s', w) \quad (5)$$

where $\mathrm{P}(t, t'|s, s', w) =$

$$\frac{\max_a \exp(w^\top \phi(t, a, t'))}{\sum_{(\bar{t},\bar{t}') \in (T,T')} \max_a \exp(w^\top \phi(\bar{t}, a, \bar{t}'))} \quad (6)$$

[4]There is a minor modification to allow nodes not to match. Any alignment link which has negative score is replaced by a zero-score link, and any zero-score link in the solution is considered a pair of unmatched nodes.

4.3 Training Set Pruning

To reduce the time and space requirements for training, we do not always use the full k-best lists. To prune the set T, we rank all the trees in T from 1 to k, according to their log likelihood under the baseline parsing model, and find the rank of the least likely pseudo-gold tree:

$$r^* = \min_{t \in \hat{T}} rank(t)$$

Finally, we restrict T based on rank:

$$T_{pruned} = \{t \in T | rank(t) \leq r^* + \epsilon\}$$

where ϵ is a free parameter of the pruning procedure. The restricted set T'_{pruned} is constructed in the same way. When training, we replace the sum over all tree pairs in (T, T') in the denominator of (6) with a sum over all tree pairs in $(T_{pruned}, T'_{pruned})$.

The parameter ϵ can be set to any value from 0 to k, with lower values resulting in more efficient training, and higher values resulting in better performance. We set ϵ by empirically determining a good speed/performance tradeoff (see §6.2).

5 Joint Selection

At test time, we have a weight vector w and so selecting optimal trees for the sentence pair (s, s') from a pair of k best lists, (T, T') is straightforward. We just find:

$$\begin{aligned}(t^*, t'^*) &= \underset{(t,t') \in (T,T')}{\arg\max} \max_a \mathrm{P}(t, a, t' | s, s', w) \\ &= \underset{(t,t') \in (T,T')}{\arg\max} \max_a w^\top \phi(t, a, t')\end{aligned}$$

Note that with no additional cost, we can also find the optimal alignment between t^* and t'^*:

$$a^* = \arg\max_a w^\top \phi(t^*, a, t'^*)$$

5.1 Test Set Pruning

Because the size of (T, T') grows as $O(k^2)$, the time spent iterating through all these tree pairs can grow unreasonably long, particularly when reranking a set of sentence pairs the size of a typical MT corpus. To combat this, we use a simple pruning technique to limit the number of tree pairs under consideration.

	Training	Dev	Test
Articles	1-270	301-325	271-300
Ch Sentences	3480	352	348
Eng Sentences	3472	358	353
Bilingual Pairs	2298	270	288

Table 1: Sentence counts from bilingual Chinese treebank corpus.

To prune the list of tree pairs, first we rank them according to the metric:

$$w_{\textsc{SourceLL}} \cdot \textsc{SourceLL} + w_{\textsc{TargetLL}} \cdot \textsc{TargetLL}$$

Then, we simply remove all tree pairs whose ranking falls below some empirically determined cutoff. As we show in §6.3, by using this technique we are able to speed up reranking by a factor of almost 20 without an appreciable loss of performance.

6 Statistical Parsing Experiments

All the data used to train the joint parsing model and to evaluate parsing performance were taken from articles 1-325 of the Chinese treebank, which all have English translations with gold-standard parse trees. The articles were split into training, development, and test sets according to the standard breakdown for Chinese parsing evaluations. Not all sentence pairs could be included for various reasons, including one-to-many Chinese-English sentence alignments, sentences omitted from the English translations, and low-fidelity translations. Additional sentence pairs were dropped from the training data because they had unambiguous parses in at least one of the two languages. Table 1 shows how many sentences were included in each dataset.

We had two training setups: rapid and full. In the rapid training setup, only 1000 sentence pairs from the training set were used, and we used fixed alignments for each tree pair rather than iterating (see §4.1). The full training setup used the iterative training procedure on all 2298 training sentence pairs.

We used the English and Chinese parsers in Petrov and Klein (2007)[5] to generate all k-best lists and as our evaluation baseline. Because our bilingual data is from the Chinese treebank, and the data

[5]Available at http://nlp.cs.berkeley.edu.

typically used to train a Chinese parser contains the Chinese side of our bilingual training data, we had to train a new Chinese grammar using only articles 400-1151 (omitting articles 1-270). This modified grammar was used to generate the k-best lists that we trained our model on. However, as we tested on the same set of articles used for monolingual Chinese parser evaluation, there was no need to use a modified grammar to generate k-best lists at test time, and so we used a regularly trained Chinese parser for this purpose.

We also note that since all parsing evaluations were performed on Chinese treebank data, the Chinese test sentences were in-domain, whereas the English sentences were very far out-of-domain for the Penn Treebank-trained baseline English parser. Hence, in these evaluations, Chinese scores tend to be higher than English ones.

Posterior word alignment probabilities were obtained from the word aligner of Liang et al. (2006) and DeNero and Klein (2007)[6], trained on approximately 1.7 million sentence pairs. For our alignment model we used an HMM in each direction, trained to agree (Liang et al., 2006), and we combined the posteriors using DeNero and Klein's (2007) soft union method.

Unless otherwise specified, the maximum value of k was set to 100 for both training and testing, and all experiments used a value of 25 as the ϵ parameter for training set pruning and a cutoff rank of 500 for test set pruning.

6.1 Feature Ablation

To verify that all our features were contributing to the model's performance, we did an ablation study, removing one group of features at a time. Table 2 shows the F_1 scores on the bilingual development data resulting from training with each group of features removed.[7] Note that though head word features seemed to be detrimental in our rapid training setup, earlier testing had shown a positive effect, so we reran the comparison using our full training setup, where we again saw an improvement when including these features.

[6] Available at http://nlp.cs.berkeley.edu.

[7] We do not have a test with the basic alignment features removed because they are necessary to compute $a_0(t, t')$.

	Baseline Parsers		
Features	Ch F_1	Eng F_1	Tot F_1
Monolingual	84.95	76.75	81.15
	Rapid Training		
Features	Ch F_1	Eng F_1	Tot F_1
All	86.37	78.92	82.91
−Hard align	85.83	77.92	82.16
−Scaled align	86.21	78.62	82.69
−Head word	**86.47**	**79.00**	**83.00**
−Span diff	86.00	77.49	82.07
−Num children	86.26	78.56	82.69
−Child labels	86.35	78.45	82.68
	Full Training		
Features	Ch F_1	Eng F_1	Tot F_1
All	**86.76**	79.41	**83.34**
−Head word	86.42	**79.53**	83.22

Table 2: Feature ablation study. F_1 on dev set after training with individual feature groups removed. Performance with individual baseline parsers included for reference.

ϵ	Ch F_1	Eng F_1	Tot F_1	Tree Pairs
15	85.78	77.75	82.05	1,463,283
20	85.88	77.27	81.90	1,819,261
25	86.37	78.92	82.91	2,204,988
30	85.97	79.18	82.83	2,618,686
40	86.10	78.12	82.40	3,521,423
50	85.95	78.50	82.50	4,503,554
100	86.28	79.02	82.91	8,997,708

Table 3: Training set pruning study. F_1 on dev set after training with different values of the ϵ parameter for training set pruning.

6.2 Training Set Pruning

To find a good value of the ϵ parameter for training set pruning we tried several different values, using our rapid training setup and testing on the dev set. The results are shown in Table 3. We selected 25 as it showed the best performance/speed tradeoff, on average performing as well as if we had done no pruning at all, while requiring only a quarter the memory and CPU time.

6.3 Test Set Pruning

We also tried several different values of the rank cutoff for test set pruning, using the full training setup

Cutoff	Ch F_1	Eng F_1	Tot F_1	Time (s)
50	86.34	79.26	83.04	174
100	86.61	79.31	83.22	307
200	86.67	79.39	83.28	509
500	86.76	79.41	83.34	1182
1000	86.80	79.39	83.35	2247
2000	86.78	79.35	83.33	4476
10,000	86.71	79.37	83.30	20,549

Table 4: Test set pruning study. F_1 on dev set obtained using different cutoffs for test set pruning.

	Joint Parsing		Oracle	
k	Ch F_1	Eng F_1	Ch F_1	Eng F_1
1	84.95	76.75	84.95	76.75
10	86.23	78.43	90.05	81.99
25	86.64	79.27	90.99	83.37
50	86.61	79.10	91.82	84.14
100	86.71	79.37	92.23	84.73
150	86.67	79.47	92.49	85.17

Table 5: Sensitivity to k study. Joint parsing and oracle F_1 obtained on dev set using different maximum values of k when generating baseline k-best lists.

and testing on the dev set. The results are in Table 4. For F_1 evaluation, which is on a very small set of sentences, we selected 500 as the value with the best speed/performance tradeoff. However, when reranking our entire MT corpus, we used a value of 200, sacrificing a tiny bit of performance for an extra factor of 2 in speed.[8]

6.4 Sensitivity to k

Since our bitext parser currently operates as a reranker, the quality of the trees is limited by the quality of the k-best lists produced by the baseline parsers. To test this limitation, we evaluated performance on the dev set using baseline k-best lists of varying length. Training parameters were fixed (full training setup with $k = 100$) and test set pruning was disabled for these experiments. The results are in Table 5. The relatively modest gains with increasing k, even as the oracle scores continue to improve, indicate that performance is limited more by the model's reliance on the baseline parsers than by search errors that result from the reranking approach.

6.5 Final Results

Our final evaluation was done using the full training setup. Here, we report F_1 scores on two sets of data. First, as before, we only include the sentence pairs from our bilingual corpus to fully demonstrate the gains made by joint parsing. We also report scores on the full test set to allow easier comparison with past work on Chinese parsing. For the latter evaluation, sentences that were not in the bilingual corpus were simply parsed with the baseline parsers. The results are in Table 6. Joint parsing improves F_1 by 2.5 points on out-of-domain English sentences and by 1.8 points on in-domain Chinese sentences; this represents the best published Chinese treebank parsing performance, even after sentences that lack a translation are taken into account.

[8]Using a rank cutoff of 200, the reranking step takes slightly longer than serially running both baseline parsers, and generating k-best lists takes slightly longer than getting 1-best parses, so in total, joint parsing takes about 2.3 times as long as monolingual parsing. With a rank cutoff of 500, total parsing time is scaled by a factor of around 3.8.

	F_1 on bilingual data only		
Parser	Ch F_1	Eng F_1	Tot F_1
Baseline	83.50	79.25	81.44
Joint	**85.25**	**81.72**	**83.52**
	F_1 on full test set		
Parser	Ch F_1	Eng F_1	Tot F_1
Baseline	82.91	78.93	81.00
Joint	**84.24**	**80.87**	**82.62**

Table 6: Final evaluation. Comparison of F_1 on test set between baseline parsers and joint parser.

7 Machine Translation

To test the impact of joint parsing on syntactic MT systems, we compared the results of training an MT system with two different sets of trees: those produced by the baseline parsers, and those produced by our joint parser. For this evaluation, we used a syntactic system based on Galley et al. (2004) and Galley et al. (2006), which extracts tree-to-string transducer rules based on target-side trees. We trained the system on 150,000 Chinese-English sentence pairs from the training corpus of Wang et al. (2007), and used a large (close to 5 billion tokens) 4-gram lan-

	Baseline	Joint	Moses
BLEU	18.7	**21.1**	18.8

Table 7: MT comparison on a syntactic system trained with trees output from either baseline monolingual parsers or our joint parser. To facilitate relative comparison, the Moses (Koehn et al., 2007) number listed reflects the default Moses configuration, including its full distortion model, and standard training pipeline.

guage model for decoding. We tuned and evaluated BLEU (Papineni et al., 2001) on separate held-out sets of sentences of up to length 40 from the same corpus. The results are in Table 7, showing that joint parsing yields a BLEU increase of 2.4.[9]

8 Conclusions

By jointly parsing (and aligning) sentences in a translation pair, it is possible to exploit mutual constraints that improve the quality of syntactic analyses over independent monolingual parsing. We presented a joint log-linear model over source trees, target trees, and node-to-node alignments between them, which is used to select an optimal tree pair from a k-best list. On Chinese treebank data, this procedure improves F_1 by 1.8 on Chinese sentences and by 2.5 on out-of-domain English sentences. Furthermore, by using this joint parsing technique to preprocess the input to a syntactic MT system, we obtain a 2.4 BLEU improvement.

Acknowledgements

We would like to thank the anonymous reviewers for helpful comments on an earlier draft of this paper and Adam Pauls and Jing Zheng for help in running our MT experiments.

References

Anthony Aue, Arul Menezes, Bob Moore, Chris Quirk, and Eric Ringger. 2004. Statistical machine translation using labeled semantic dependency graphs. In *TMI*.

Ann Bies, Martha Palmer, Justin Mott, and Colin Warner. 2007. English chinese translation treebank v 1.0. Web download. LDC2007T02.

Daniel M. Bikel and David Chiang. 2000. Two statistical parsing models applied to the chinese treebank. In *Second Chinese Language Processing Workshop*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *ACL*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

John DeNero and Dan Klein. 2007. Tailoring word alignments to syntactic machine translation. In *ACL*.

Yuan Ding and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *ACL*.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *HLT-NAACL*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *COLING-ACL*.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *HLT-NAACL*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL*.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *HLT-NAACL*.

Yuval Marton and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrase-based translation. In *ACL*.

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. 2003. Syntax for statistical machine translation. Technical report, CLSP, Johns Hopkins University.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. Research report, IBM. RC22176.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *HLT-NAACL*.

[9]Note that all numbers are single-reference BLEU scores and are not comparable to multiple reference scores or scores on other corpora.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal smt. In *ACL*.

Libin Shen, Jinxi Xu, and Ralph Weishedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *ACL*.

David A. Smith and Noah A. Smith. 2004. Bilingual parsing with factored estimation: using english to parse korean. In *EMNLP*.

Leslie G. Valiant. 1979. The complexity of computing the permanent. In *Theoretical Computer Science 8*.

Wen Wang, Andreas Stolcke, and Jing Zheng. 2007. Reranking machine translation hypotheses with structured and web-based language models. In *IEEE ASRU Workshop*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404.

Nianwen Xue, Fu-Dong Chiou, and Martha Palmer. 2002. Building a large-scale annotated chinese corpus. In *COLING*.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *ACL*.

Hao Zhang, Chris Quirk, Robert C. Moore, and Daniel Gildea. 2008. Bayesian learning of non-compositional phrases with synchronous parsing. In *ACL*.

Andreas Zollmann, Ashish Venugopal, Stephan Vogel, and Alex Waibel. 2006. The cmu-aka syntax augmented machine translation system for iwslt-06. In *IWSLT*.

Automatic Prediction of Parser Accuracy

Sujith Ravi and Kevin Knight
University of Southern California
Information Sciences Institute
Marina del Rey, California 90292
{sravi,knight}@isi.edu

Radu Soricut
Language Weaver, Inc.
4640 Admiralty Way, Suite 1210
Marina del Rey, California 90292
rsoricut@languageweaver.com

Abstract

Statistical parsers have become increasingly accurate, to the point where they are useful in many natural language applications. However, estimating parsing accuracy on a wide variety of domains and genres is still a challenge in the absence of gold-standard parse trees.

In this paper, we propose a technique that automatically takes into account certain characteristics of the domains of interest, and accurately predicts parser performance on data from these new domains. As a result, we have a cheap (no annotation involved) and effective recipe for measuring the performance of a statistical parser on any given domain.

1 Introduction

Statistical natural language parsers have recently become more accurate and more widely available. As a result, they are being used in a variety of applications, such as question answering (Hermjakob, 2001), speech recognition (Chelba and Jelinek, 1998), language modeling (Roark, 2001), language generation (Soricut, 2006) and, most notably, machine translation (Charniak et al., 2003; Galley et al., 2004; Collins et al., 2005; Marcu et al., 2006; Huang et al., 2006; Avramidis and Koehn, 2008). These applications are employed on a wide range of domains and genres, and therefore the question of how accurate a parser is on the domain and genre of interest becomes acute. Ideally, one would want to have available a recipe for precisely answering this question: "given a parser and a particular domain of interest, how accurate are the parse trees produced?" The only recipe that is implicitly given in the large literature on parsing to date is to have human annotators build parse trees for a sample set from the domain of interest, and consequently use them to compute a PARSEVAL (Black et al., 1991) score that is indicative of the intrinsic performance of the parser. Given the wide range of domains and genres for which NLP applications are of interest, combined with the high expertise required from human annotators to produce parse tree annotations, this recipe is, albeit precise, too expensive. The other recipe that is currently used on a large scale is to measure the performance of a parser on existing treebanks, such as WSJ (Marcus et al., 1993), and assume that the accuracy measure will carry over to the domains of interest. This recipe, albeit cheap, cannot provide any guarantee regarding the performance of a parser on a new domain, and, as experiments in this paper show, can give wrong indications regarding important decisions for the design of NLP systems that use a syntactic parser as an important component.

This paper proposes another method for measuring the performance of a parser on a given domain that is both cheap and effective. It is a fully automated procedure (no expensive annotation involved) that uses properties of both the domain of interest and the domain on which the parser was trained in order to measure the performance of the parser on the domain of interest. It is, in essence, a solution to the following prediction problem:

Input: (1) a statistical parser and its training data, (2) some chunk of text from a new domain or genre

Output: an estimate of the accuracy of the parse trees produced for that chunk of text

Accurate estimations for this prediction problem will allow a system designer to make the right decisions for the given domain of interest. Such decisions include, but are not restricted to, the choice of the parser, the choice of the training data, the choice of how to implement various components such as the treatment of unknown words, etc. Altogether, a correct estimation of the impact of such decisions on the resulting parse trees can guide a system designer in a hill-climbing scenario for which an extrinsic metric (such as the impact on the overall quality of the system) is usually too expensive to be employed often enough. To provide an example, a machine translation engine that requires parse trees as training data in order to learn syntax-based translation rules (Galley et al., 2006) needs to employ a syntactic parser as soon as the training process starts, but it can take up to hundreds and even thousands of CPU hours (for large training data sets) to train the engine before translations can be produced and measured. Although a real estimate of the impact of a parser design decision in this scenario can only be gauged from the quality of the translations produced, it is impractical to create such estimates for each design decision. On the other hand, estimates using the solution proposed in this paper can be obtained fast, before submitting the parser output to a costly training procedure.

2 Related Work and Experimental Framework

There have been previous studies which explored the problem of automatically predicting the task difficulty for various NLP applications. (Albrecht and Hwa, 2007) presented a regression based method for developing automatic evaluation metrics for machine translation systems without directly relying on human reference translations. (Hoshino and Nakagawa, 2007) built a computer-adaptive system for generating questions to teach English grammar and vocabulary to students, by predicting the difficulty level of a question using various features. There have been a few studies of English parser accuracy in domains/genres other than WSJ (Gildea, 2001; Bacchiani et al., 2006; McClosky et al., 2006), but in order to make measurements for such studies, it is necessary to have gold-standard parses in the non-WSJ domain of interest.

Gildea (2001) studied how well WSJ-trained parsers do on the Brown corpus, for which a gold standard exists. He looked at sentences with 40 words or less. (Bacchiani et al., 2006) carried out a similar experiment on sentences of all lengths, and (McClosky et al., 2006) report additional results. The table below shows results from our own measurements of Charniak parser[1] (Charniak and Johnson, 2005) accuracy (F-measure on sentences of all lengths), which are consistent with these studies. For the Brown corpus, the test set was formed from every tenth sentence in the corpus (Gildea, 2001).

Training Set	**Test Set**	**Sent. count**	**Charniak accuracy**
WSJ sec. 02-21 (39,832 sent.)	WSJ sec. 24	1308	90.48
	WSJ sec. 23	2343	91.13
	Brown-test	2186	86.34

Here we investigate algorithms for predicting the accuracy of a parser $\mathcal{P}$ on sentences, chunks of sentences, and whole corpora. We also investigate and contrast several scenarios for prediction: (1) the predictor looks at the input text only, (2) the predictor looks at the input text and the output parse trees of $\mathcal{P}$, and (3) the predictor looks at the input text, the output parse trees of $\mathcal{P}$, and the outputs of other programs, such as the output parse trees of a different parser $\mathcal{P}_{ref}$ used as a reference. Under none of these scenarios is the predictor allowed to look at gold-standard parses in the new domain/genre.

The intuition behind what we are trying to achieve here can be compared to an analogous task—trying to assess the performance of a median student from a math class on a given test, without having access to the answer sheet. Looking at the test only, we could probably tell whether the test looks hard or not, and therefore whether the student will do well or not. Looking at the student's answers will likely give us an even better idea of the performance. Finally, the answers of a second student with similar proficiency will provide even better clues: if the students agree on every answer, then they probably both did well, but if they disagree frequently, then they (and hence our student) probably did not do as well.

Our first experiments are concerned with validating the idea itself: can a predictor be trained such

[1]Downloaded from *ftp.cs.brown.edu/pub/nlparser/reranking-parserAug06.tar.gz* in February, 2007.

that it predicts the same F-scores as the ones obtained using gold-trees? We first validate this using the WSJ corpus itself, by dividing the WSJ treebank into several sections:

1. *Training* (WSJ section 02-21). The parser $\mathcal{P}$ is trained on this data.

2. *Development* (WSJ section 24). We use this data for training our predictor.

3. *Test* (WSJ section 23). We use this data for measuring our predictions. For each test sentence, we compute (1) the PARSEVAL F-measure score using the test gold standard, and (2) our predicted F-measure. We report the correlation coefficient (r) between the actual F-scores and our predicted F-scores. We will also use a root-mean-square error (rms error) metric to compare actual and predicted F-scores.

Section 3 describes the features used by our predictor. Given these features, as well as actual F-scores computed for the development data, we use supervised learning to set the feature weights. To this end, we use SVM-Regression[2] (Smola and Schoelkopf, 1998) with an RBF kernel, to learn the feature weights and build our predictor system.[3] We validate the accuracy of the predictor trained in this fashion on both WSJ (Section 4) and the Brown corpus (Section 5).

3 Features Used for Predicting Parser Accuracy

3.1 Text-based Features

One hypothesis we explore is that (all other things being equal) longer sentences are harder to parse correctly than shorter sentences. When exposed to the development set, SVM-Regression learns weights to best predict F-scores using the values for this feature corresponding to each sentence in the corpus.

Does the predicted F-score correlate with actual F-score on a sentence by sentence basis? There was a positive but weak correlation:

Feature set	**dev (r)**	**test (r)**
Length	0.13	0.19

Another hypothesis is that the parser performance is influenced by the number of UNKNOWN words in the sentence to be parsed, i.e., the number of words in the test sentence that were never seen before in the training set. Training the predictor with this feature produces a positive correlation, slightly weaker compared to the Length feature.

Feature set	**dev (r)**	**test (r)**
UNK	0.11	0.11

Unknown words are not the only ones that can influence the performance of a parser. Rare words, for which statistical models do not have reliable estimates, are also likely to impact parsing accuracy. To test this hypothesis, we add a language model perplexity–based (LM-PPL) feature. We extract the yield of the training trees, on which we train a trigram language model.[4] We compute the perplexity of each test sentence with respect to this language model, and use it as feature in our predictor system. Note that this feature is meant as a refinement of the previous UNK feature, in the sense that perplexity numbers are meant to signal the occurrence of unknown words, as well as rare (from the training data perspective) words. However, the correlation we observe for this feature is similar to the correlation observed for the UNK feature, which seems to suggest that the smoothing techniques used by the parsers employed in these experiments lead to correct treatment of the rare words.

Feature set	**dev (r)**	**test (r)**
LM-PPL	0.11	0.10

We also look at the possibility of automatically detecting certain "cue" words that are appropriate for our prediction problem. That is, we want to see if we can detect certain words that have a discriminating power in deciding whether parsing a sentence that contains them is difficult or easy. To this end, we use a subset of the development data, which contains the 200 best-parsed and 200 worst-parsed sentences (based on F-measure scores). For each word in the development dataset, we compute the information gain (IG) (Yang and Pedersen, 1997) score for that word with respect to the best/worst parsed

[2]Weka software (http://www.cs.waikato.ac.nz/ml/weka/)

[3]We compared a few regression algorithms like SVM-Regression (using different kernels and parameter settings) and Multi-Layer Perceptron (neural networks) – we trained the algorithms separately on dev data and picked the one that gave the best cross-validation accuracy (F-measure).

[4]We trained using the SRILM language modeling toolkit, with default settings.

dataset. These words are then ranked by their IG scores, and the top 100 words are included as lexical features in our predictor system. As expected, the correlation on the development set is quite high (given that these lexical cues are extracted from this particular set), but a positive correlation holds for the test set as well.

Feature set	dev (r)	test (r)
lexCount100	0.43	0.18

3.2 Parser $\mathcal{P}$–based Features

Besides exploiting the information present in the input text, we can also inspect the output tree of the parser $\mathcal{P}$ for which we are interested in predicting accuracy. We create a rootSYN feature based on the syntactic category found at the root of the output tree ("is it S?", "is it FRAG?"). We also create a puncSYN feature based on the number of words labeled as punctuation tags (based on the intuition that heavy use of punctuation can be indicative of the difficulty of the input sentences), and a labelSYN feature in which we bundled together information regarding the number of internal nodes in the parse tree output that have particular labels ("how many nodes are labeled with PP?"). In our predictor, we use 72 such labelSYN features corresponding to all the syntactic labels found in the parse tree output for the development set. The test set correlation given by the rootSYN and the labelSYN features are higher than some of the text-based features, whereas the puncSYN feature seems to have little discriminative power.

Feature set	dev (r)	test (r)
rootSYN	0.21	0.17
puncSYN	0.09	0.01
labelSYN	0.33	0.28

3.3 Reference Parser $\mathcal{P}_{ref}$–based Features

In addition to the text-based features and parser $\mathcal{P}$–based features, we can bring in an additional parser $\mathcal{P}_{ref}$ whose output is used as a reference against which the output of parser $\mathcal{P}$ is measured. For the reference parser feature, our goal is to measure how similar/different are the results from the two parsers. We find that if the parses are similar, they are more likely to be right. In order to compute similarity, we can compare the constituents in the two parse trees from $\mathcal{P}$ and $\mathcal{P}_{ref}$, and see how many constituents match. This is most easily accomplished by considering $\mathcal{P}_{ref}$ to be a "gold standard" (even though it is not necessarily a correct parse) and computing the F-measure score of parser $\mathcal{P}$ against $\mathcal{P}_{ref}$. We use this F-measure score as a feature for prediction.

For the experiments presented in this section we use as $\mathcal{P}_{ref}$, the parser from (Bikel, 2002). Intuitively, the requirement for choosing parser $\mathcal{P}_{ref}$ in conjunction with parser $\mathcal{P}$ seems to be that they are different enough to produce non-identical trees when presented with the same input, and at the same time to be accurate enough to produce reliable parse trees. The choice of $\mathcal{P}$ as (Charniak and Johnson, 2005) and $\mathcal{P}_{ref}$ as (Bikel, 2002) fits this bill, but many other choices can be made regarding $\mathcal{P}_{ref}$, such as (Klein and Manning, 2003; Petrov and Klein, 2007; McClosky et al., 2006; Huang, 2008). We leave the task of creating features based on the consensus of multiple parsers as future work.

The correlation given by the reference parser–based feature $\mathcal{P}_{ref}$ on the test set is the highest among all the features we explored.

Feature set	dev (r)	test (r)
$\mathcal{P}_{ref}$	0.40	0.36

3.4 The Aggregated Power of Features

The table below lists all the individual features we have described in this section, sorted according to the correlation value obtained on the test set.

Feature set	dev (r)	test (r)
$\mathcal{P}_{ref}$	0.40	0.36
labelSYN	0.33	0.28
lexCount500	0.56	0.23
lexBool500	0.58	0.20
lexCount1000	0.67	0.20
lexBool1000	0.58	0.20
Length	0.13	0.19
lexCount100	0.43	0.18
lexBool100	0.43	0.18
rootSYN	0.21	0.17
UNK	0.11	0.11
LM-PPL	0.11	0.10
puncSYN	0.09	0.01

Note how the lexical features tend to over-fit the development data—the words were specifically chosen for their discriminating power on that particular set. Hence, adding more lexical features to the predictor system improves the correlation on development (due to over-fitting), but it does not produce consistent improvement on the test set. However,

Method (using 3 features: Length, UNK, $\mathcal{P}_{ref}$)	**# of random restarts**	**dev (r)**
SVM Regression		**0.42**
Maximum Correlation Training (MCT)	1	0.138
	5	0.136
	10	0.166
	25	0.178
	100	0.232
	1000	0.27
	10,000	0.401

Table 1: Comparison of correlation (r) obtained using MCT versus SVM-Regression on development corpus.

there is some indication that the counts of the lexical features are important, and count-based lexical features tend to have similar or better performance compared to their boolean-based counterparts.

Since these features measure different but overlapping pieces of the information available, it is to be expected that some of the feature combinations would provide better correlation that the individual features, but the gains are not strictly additive. By taking the individual features that provide the best discriminative power, we are able to get a correlation score of 0.42 on the test set.

Feature set	**dev (r)**	**test (r)**
$\mathcal{P}_{ref}$ + labelSYN + Length + lexCount100 + rootSYN + UNK + LM-PPL	0.55	0.42

3.5 Optimizing for Maximum Correlation

If our goal is to obtain the highest correlations with the F-score measure, is SVM regression the best method? Liu and Gildea (2007) recently introduced Maximum Correlation Training (MCT), a search procedure that follows the gradient of the formula for correlation coefficient (r). We implemented MCT, but obtained no better results. Moreover, it required many random re-starts just to obtain results comparable to SVM regression (Table 1).

4 Predicting Accuracy on Multiple Sentences

The results for the scenario presented in Section 3 are encouraging, but other scenarios are also important from a practical perspective. For instance, we are interested in predicting the performance of a particular parser not on a sentence-by-sentence basis, but for a representative chunk of sentences from the new domain. In order to predict the F-measure on multiple sentences, we modify our feature set to generate information on a whole chunk of sentences rather than a single sentence. Predicting the correlation at chunk level is, not unexpectedly, an easier problem than predicting correlation at sentence level, as the results in the first two columns of Table 2 show.

Sentences in chunk (n)	**WSJ-test (r)**	**WSJ-test (rms error)**
1	0.42	0.098
20	0.61	0.026
50	0.62	0.019
100	0.69	0.015
500	0.79	0.011

Table 2: Performance of predictor on n-sentence chunks from WSJ-test (Correlation and rms error between actual/predicted accuracies).

For 100-sentence chunks, we also plot the predicted accuracies versus actual accuracies for the WSJ-test set in Figure 1. This scatterplot brings to light an artifact of using correlation metric (r) for evaluating our predictor's performance. Although our objective is to improve correlation between actual and predicted F-scores, the correlation metric (r) does not tell us directly how well the predictor is doing. In Figure 1, the system predicts that on an average, most sentence chunks can be parsed with an accuracy of 0.9085 (which is the mean predicted F-score on WSJ-test). But the range of predictions from our system [0.89,0.92] is smaller than the actual F-score range [0.86,0.95]. Hence, even though the correlation scores are high, this does not necessarily mean that our predictions are on target. An additional metric, *root-mean-square (rms) error*, which measures the distance between actual and predicted F-measures, can be used to gauge the quality of our predictions. For a particular chunk-size, lowering the *rms error* translates into aligning the points of a scatterplot as the one in Figure 1, closer to the x=y line, implying that the predictor is getting better at exactly predicting the F-score values. The third column in Table 2 shows the rms error for our predictor at different chunk sizes. The results using this metric also show that the prediction problem becomes easier as the chunk size increases.

Assuming that we have the test set of WSJ section 23, but without the gold-standard trees, how can we get an approximation for the overall accuracy of a parser $\mathcal{P}$ on this test set? One possibility, which we use here as a baseline, is to compute the F-score on a set for which we do have gold-standard trees. If we use our development set (WSJ section

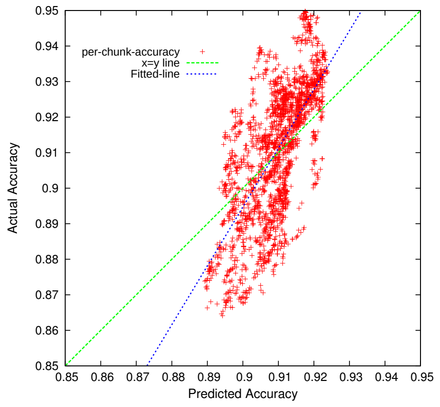


Figure 1: Plot showing Actual vs. Predicted accuracies for WSJ-test (100-sentence chunks). Each plot point represents a 100-sentence chunk. *(rms error = 0.015)*

System	F-measure
Charniak F-measure on WSJ-dev (baseline)	90.48 (f_d)
Predictor (feature weights set with WSJ-dev)	**90.85** (f_p)
Actual Charniak accuracy	91.13 (f_t)

Table 3: Comparing Charniak parser accuracy (from different systems) on entire WSJ-test corpus

24) for this purpose, and (Charniak and Johnson, 2005) as the parser $\mathcal{P}$, the baseline is an F-score of 90.48 (f_d), which is the actual Charniak parser accuracy on WSJ section 24. Instead, if we run our predictor on the test set (a single chunk containing all the sentences in the test set), it predicts an F-score of 90.85 (f_p). These two predictions are listed as the first two rows in Table 3. Of course, having the actual gold-standard trees for WSJ section 23 helps us decide which prediction is better: the actual accuracy of the Charniak parser on WSJ section 23 is an F-score of 91.13 (f_t), which makes our prediction better than the baseline.

4.1 Shifting Predictions to Match Actual Accuracy

We correctly predict (in Table 3) that the WSJ-test is easier to parse than the WSJ-dev (90.85 > 90.48). However, our predictor is too conservative—the WSJ-test is actually even easier to parse (91.13 > 90.85). We can fix this by shifting the mean predicted F-score (which is equal to f_p) further away from the dev F-measure (f_d), and closer to the actual F-measure (f_t). This is achieved by shifting all the individual predictions by a certain amount as shown below.

Let p be an individual prediction from our system.

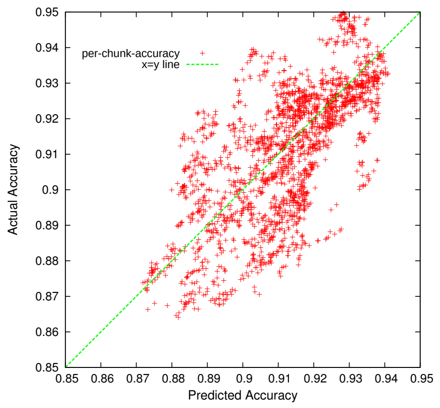


Figure 2: Plot showing Actual vs. Adjusted Predicted accuracies *(shifting with $\alpha = 0.757$, skewing with $\beta = 1.0$)* for WSJ-test (100-sentence chunks). *(rms error = 0.014)*

The shifted prediction p' is given by:

$$p' = p + \alpha(f_p - f_d) \qquad (1)$$

We can tune α to make the new mean prediction (f'_p) to be equal to the actual F-measure (f_t).

$$f'_p = f_p + \alpha(f_p - f_d) \qquad (2)$$

$$\alpha = \frac{f_t - f_p}{f_p - f_d} \qquad (3)$$

Using the F-score values from Table 3, we get an $\alpha = 0.757$ and an exact prediction of 91.13. Of course, this is because we tune on test, so we need to validate this idea on a new test set to see if it leads to improved predictions (Section 5).

4.2 Skewing to Widen Prediction Range

Our predictor is also too conservative about its *distribution* (see Figure 1). It knows (roughly) which chunks are easier to parse and which are harder, but its range of predictions is lower than the range of actual F-measure scores.

We can skew individual predictions so that sentences predicted to be easy are re-predicted to be even easier (and those that are hard to be even harder). For each prediction p' (from Equation 1), we compute

$$p'' = p' + \beta(p' - f'_p) \qquad (4)$$

We simply set β to 1.0, doubling the distance of each prediction p' (in Equation 1) from the (adjusted) mean prediction f'_p, to obtain the skewed prediction p''.

Figure 2 shows how the points representing 100-sentence chunks in Figure 1 look after the predictions have been shifted ($\alpha = 0.757$) and skewed ($\beta = 1.0$). These two operations have the desired effect of changing the range of predictions from [0.89,0.92] to [0.87,0.94], much closer to the actual

Sentences in chunk (n)	WSJ-test (rms error)	Brown-test Prediction (rms error)	Brown-test Adjusted Prediction (rms error)
1	0.098	0.129	0.139
20	0.026	0.039	0.036
50	0.019	0.032	0.029
100	0.015	0.025	0.020
500	0.011	0.038	0.024

Table 4: Performance of predictor on n-sentence chunks from WSJ-test and Brown-test (rms error between actual/predicted accuracies).

range of [0.86,0.95]. The points in the new plot (Figure 2) also align closer to the "x=y" line than in the original graph (Figure 1). The rms error also drops from 0.015 to 0.014 (7% relative reduction), showing that the predictions have improved.

Since we use the WSJ-test corpus to tune the parameter values for shifting and skewing, we need to apply our predictor on a different test set to see if we get similar improvements by using these techniques, which we do in the next section.

5 Predicting Accuracy on the Brown Corpus

The Brown corpus represents a genuine challenge for our predictor, as it presents us with the opportunity to test the performance of our predictor in an out-of-domain scenario. Our predictor, trained on WSJ data, is now employed to predict the performance of a WSJ-trained parser $\mathcal{P}$ on the Brown-test corpus. As in the previous experiments, we use (Charniak and Johnson, 2005) trained on WSJ sections 02-21 as parser $\mathcal{P}$. The feature weights for our predictor are again trained on section 24 of WSJ, and the shifting and skewing parameters ($\alpha = 0.757$, $\beta = 1.0$) are determined using section 23 of WSJ.

The results on the Brown-test, both the original predictions and after they have been adjusted (shifted/skewed), are shown in Table 4, at different level of chunking. For chunks of size n > 1, the shifting and skewing techniques help in lowering the rms error. On 100-sentence chunks from the Brown test, shifting and skewing ($\alpha = 0.757$, $\beta = 1.0$) leads to a 20% relative reduction in the rms error.

In a similar vein with the evaluation done in Section 4, we are interested in estimating the overall accuracy of a WSJ-trained parser $\mathcal{P}$ given an out-of-domain set such as the Brown test set (for which, at least for now, we do not have access to gold-standard trees). If we use (Charniak and Johnson, 2005) as parser $\mathcal{P}$, a cheap and readily-available answer is to approximate the performance using the Charniak parser performance on WSJ section 23, which has an F-score of 91.13. Another cheap and readily-available answer is to take the Charniak parser performance on WSJ section 24 with an F-score of 90.48. Table 5 lists these baselines, along with the prediction made by our system when using a single chunk containing all the sentences in the Brown test set (both base predictions and adjusted predictions, i.e. shifting using $\alpha = 0.757$). Again, having gold-standard trees for the Brown test set helps us decide which prediction is better. Our predictions are much closer to the actual Charniak parser performance on the Brown-test set, with the adjusted prediction at 86.96 compared to the actual F-score of 86.34.

System	F-measure
Baseline1 (F-measure on WSJ sec. 23)	91.13
Baseline2 (F-measure on WSJ sec. 24)	90.48
Predictor (base)	88.48
Adjusted Predictor (shifting using $\alpha = 0.757$)	**86.96**
Actual accuracy	86.34

Table 5: Charniak parser accuracy on entire Brown-test corpus

6 Ranking Parser Performance

One of the main goals for computing F-score figures (either by traditional PARSEVAL evaluation against gold standards or by methods such as the one proposed in this paper) is to compare parsing accuracy when confronted with a choice between various parser deployments. Not only are there many parsing techniques available (Collins, 2003; Charniak and Johnson, 2005; Petrov and Klein, 2007; McClosky et al., 2006; Huang, 2008), but recent annotation efforts in providing training material for statistical parsing (LDC, 2005; LDC, 2006a; LDC, 2006b; LDC, 2006c; LDC, 2007) have compounded the difficulty of the choices ("Do I parse using parser X?", "Do I train parser X using the treebank Y or Z?"). In this section, we show how our predictor can provide guidance when dealing with some of these choices, namely the choice of the training material to use with a statistical parser, prior to its application in an NLP task.

For the experiments reported in this paper, we use as parser $\mathcal{P}$, our in-house implementation of the Collins parser (Collins, 2003), to which various speed-related enhancements (Goodman, 1997) have been applied. This choice has been made to better reflect a scenario in which parser $\mathcal{P}$ would be used in a data-intensive application such as syntax-driven machine translation, in which the parser must be able to run through hundreds of millions of training words in a timely manner. We use the more accurate, but slower Charniak parser (Charniak and Johnson, 2005) as the reference parser $\mathcal{P}_{ref}$ in our predictor (see Section 3.3). In order to predict the Collins-style parser behavior on the ranking task, we use the same predictor model (including feature weights and adjustment parameters) that was used for predicting Charniak parser behavior on the Brown corpus (Section 5).

We compare three training scenarios that make for three different parsers:

(1) P_{WSJ} - trained on sections 02-21 of WSJ.

(2) P_{News} - trained on the union of the English Chinese Translation Treebank (LDC, 2007) (news stories from Xinhua News Agency translated from Chinese into English) and the English Newswire Translation Treebank (LDC, 2005; LDC, 2006a; LDC, 2006b; LDC, 2006c) (An-Nahar new stories translated from Arabic into English).

(3) $\text{P}_{WSJ-News}$ - trained on the union of all the above training material.

When comparing the performance of these three parsers on a development set from WSJ (section 0), we get the following F-scores.[5]

Parser	**WSJ (sec. 0) Accuracy (F-scores)**
P_{WSJ}	88.25
P_{News}	83.00
$\text{P}_{WSJ-News}$	88.00

Consider now that we are interested in comparing the parsing accuracy of these parsers on a domain completely different from WSJ. The ranking P_{WSJ}>$\text{P}_{WSJ-News}$>P_{News}, given by the evaluation above, provides some guidance, but is this guidance accurate? The intuition here is that the information that we already have about the new domain of interest (which implicitly appears in texts extracted from this domain), can be used to better guide this decision. Our predictor is able to capitalize on this information, and provide domain-informed guidance for choosing the most accurate parser to use with the new data, which in this case relates to choosing the best training strategy for the parser $\mathcal{P}$. If we consider as our domain of interest, news stories from Xinhua News Agency, then using our predictor on a chunk of 1866 sentences from this domain gives the F-scores shown in the second column of Table 6.

[5]Because of tokenization differences between the different treebanks involved in these experiments, we have to adopt a tokenization scheme different from the one used in the Penn Treebank, and therefore the F-scores, albeit in the same range, are not directly comparable with the ones in the parsing literature.

Parser	**Xinhua News Prediction (F-scores)**	**Xinhua News Accuracy (F-scores)**
P_{WSJ}	85.1	79.14
P_{News}	87.0	84.84
$\text{P}_{WSJ-News}$	89.4	85.14

Table 6: Performance of predictor on the Xinhua News domain, compared with actual F-scores.

As with the previous experiments, we can compute the actual PARSEVAL F-scores (using gold-standard) for this particular 1866-sentence test set, as it happens to be part of the English Chinese Translation Treebank (LDC, 2007). These F-score figures are shown in the third column of Table 6. As these results show, for this particular domain the correct ranking is $\text{P}_{WSJ-News}$>P_{News}>P_{WSJ}, which is exactly the ranking predicted by our method, without the aid of gold-standard trees.

We observe that even though the system predicts the ranking correctly, the predictions in the Xinhua News domain might not be as accurate in comparison to the predictions on Brown corpus (predicted F-score = 86.96, actual F-score = 86.34). One possible reason for this lower accuracy is that we use the same prediction model without optimizing for the particular parser on which we wish to make predictions. Still, the model was able to make distinctions between multiple parsers for the ranking task correctly, and decide the best parser to use with the given data. We believe this to be useful in typical NLP applications which use parsing as a component, and where making the right choice between different parsers can affect the end-to-end accuracy of the system.

7 Conclusion

The steady advances in statistical parsing over the last years have taken this technology to the point where it is accurate enough to be useful in a variety of natural language applications. However, due to large variations in the characteristics of the domains for which these applications are developed, estimating parsing accuracy becomes more involved than simply taking for granted accuracy estimates done on a certain well-studied domain, such as WSJ. As the results in this paper show, it is possible to take into account these variations in the domain characteristics (encoded in our predictor as text-based, syntax-based, and agreement-based features)—to make better predictions about the accuracy of certain statistical parsers (and under different training scenarios), instead of relying on accuracy estimates done on a standard domain. We have provided a mechanism to incorporate these domain variations for making predictions about parsing accuracy, without the costly requirement of creating human annotations for each of the domains of interest. The experiments shown in the paper were limited to readily available statistical parsers (which are widely deployed in a number of applications), and certain domains/genres (because of ready access to gold-standard data on which we could verify predictions). However, the features we use in our predictor are independent of the particular type of parser or domain, and the same technique could be applied for making predictions on other parsers as well.

There are many avenues for future work opened up by the work presented here. The accuracy of the predictor can be further improved by incorporating more complex syntax-based features and multiple-agreement features. Moreover, rather than predicting an intrinsic metric such as the PARSEVAL F-score, the metric that the predictor learns to predict can be chosen to better fit the final metric on which an end-to-end system is measured, in the style of (Och, 2003). The end-result is a finely-tuned tool for predicting the impact of various parser design decisions on the overall quality of a system.

8 Acknowledgements

We wish to acknowledge our colleagues at ISI, who provided useful suggestions and constructive criticism on this work. We are also grateful to all the reviewers for their detailed comments. This work was supported in part by NSF grant IIS-0428020.

References

Joshua Albrecht and Rebecca Hwa. 2007. Regression for sentence-level mt evaluation with pseudo references. In *Proc. of ACL.*

Eleftherios Avramidis and Philipp Koehn. 2008. Enriching morphologically poor languages for statistical machine translation. In *Proc. of ACL.*

Michiel Bacchiani, Michael Riley, Brian Roark, and Richard Sproat. 2006. MAP adaptation of stochastic grammars. *Computer Speech & Language*, 20(1).

Daniel M. Bikel. 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. In *Proc. of HLT.*

E. Black, S. Abney, D. Flickinger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of english grammars. In *Proc. of Speech and Natural Language Workshop.*

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proc. of ACL.*

Eugene Charniak, Kevin Knight, and Kenji Yamada. 2003. Syntax-based language models for statistical machine translation. In *Proc. of MT Summit IX. IAMT.*

Ciprian Chelba and Frederick Jelinek. 1998. Exploiting syntactic structure for language modeling. In *Proc. of ACL.*

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proc. of ACL.*

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4).

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proc. of HLT/NAACL.*

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. of ACL.*

Daniel Gildea. 2001. Corpus variation and parser performance. In *Proc. of EMNLP.*

Joshua Goodman. 1997. Global thresholding and multiple-pass parsing. In *Proc. of EMNLP.*

Ulf Hermjakob. 2001. Parsing and question classification for question answering. In *Proc. of ACL Workshop on Open-Domain Question Answering.*

Ayako Hoshino and Hiroshi Nakagawa. 2007. A cloze test authoring system and its automation. In *Proc. of ICWL.*

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proc. of AMTA*.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proc. of ACL*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proc. of ACL*.

LDC. 2005. English newswire translation treebank. Linguistic Data Consortium, Catalog number LDC2005E85.

LDC. 2006a. English newswire translation treebank. Linguistic Data Consortium, Catalog number LDC2006E36.

LDC. 2006b. GALE Y1 Q3 release - English translation treebank. Linguistic Data Consortium, Catalog number LDC2006E82.

LDC. 2006c. GALE Y1 Q4 release - English translation treebank. Linguistic Data Consortium, Catalog number LDC2006E95.

LDC. 2007. English chinese translation treebank. Linguistic Data Consortium, Catalog number LDC2007T02.

Ding Liu and Daniel Gildea. 2007. Source-language features and maximum correlation training for machine translation evaluation. In *Proc. of NAACL-HLT*.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. Spmt: Statistical machine translation with syntactified target language phraases. In *Proc. of EMNLP*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2).

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Reranking and self-training for parser adaptation. In *Proc. of COLING-ACL*.

Franz Joseph Och. 2003. Minimum error rate training in machine translation. In *Proc. of ACL*.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proc. of HLT/NAACL*.

Brian Roark. 2001. Probabilistic top-down parsing and language modelling. *Computational Linguistics*, 27(2).

A.J. Smola and B. Schoelkopf. 1998. A tutorial on support vector regression. *NeuroCOLT2 Technical Report NC2-TR-1998-030*.

Radu Soricut. 2006. *Natural Language Generation using an Information-Slim Representation*. Ph.D. thesis, University of Southern California,.

Y. Yang and J. Pedersen. 1997. A comparative study on feature selection in text categorization. In *Proc. of ICML*.

A Structured Vector Space Model for Word Meaning in Context

Katrin Erk
Department of Linguistics
University of Texas at Austin
katrin.erk@mail.utexas.edu

Sebastian Padó
Department of Linguistics
Stanford University
pado@stanford.edu

Abstract

We address the task of computing vector space representations for the meaning of word occurrences, which can vary widely according to context. This task is a crucial step towards a robust, vector-based compositional account of sentence meaning. We argue that existing models for this task do not take syntactic structure sufficiently into account.

We present a novel *structured vector space* model that addresses these issues by incorporating the selectional preferences for words' argument positions. This makes it possible to integrate syntax into the computation of word meaning in context. In addition, the model performs at and above the state of the art for modeling the contextual adequacy of paraphrases.

1 Introduction

Semantic spaces are a popular framework for the representation of word meaning, encoding the meaning of lemmas as high-dimensional vectors. In the default case, the components of these vectors measure the co-occurrence of the lemma with context features over a large corpus. These vectors are able to provide a robust model of *semantic similarity* that has been used in NLP (Salton et al., 1975; McCarthy and Carroll, 2003; Manning et al., 2008) and to model experimental results in cognitive science (Landauer and Dumais, 1997; McDonald and Ramscar, 2001). Semantic spaces are attractive because they provide a model of word meaning that is independent of dictionary senses and their much-discussed problems (Kilgarriff, 1997; McCarthy and Navigli, 2007).

In a default semantic space as described above, each vector represents one *lemma*, averaging over all its possible usages (Landauer and Dumais, 1997; Lund and Burgess, 1996). Since the meaning of words can vary substantially between occurrences (e.g., for polysemous words), the next necessary step is to characterize the meaning of individual *words in context*.

There have been several approaches in the literature (Smolensky, 1990; Schütze, 1998; Kintsch, 2001; McDonald and Brew, 2004; Mitchell and Lapata, 2008) that compute meaning in context from lemma vectors. Most of these studies phrase the problem as one of vector composition: The meaning of a target occurrence a in context b is a single new vector c that is a function (for example, the centroid) of the vectors: $c = a \odot b$.

The context b can consist of as little as one word, as shown in Example (1). In (1a), the meaning of *catch* combined with *ball* is similar to *grab*, while in (1b), combined with *disease*, it can be paraphrased by *contract*. Conversely, verbs can influence the interpretation of nouns: In (1a), *ball* is understood as a spherical object, and in (1c) as a dancing event.

(1) a. catch a ball
  b. catch a disease
  c. attend a ball

In this paper, we argue that models of word meaning relying on this procedure of vector composition are limited both in their scope and scalability. The underlying shortcoming is a failure to consider *syntax* in two important ways.

The syntactic relation is ignored. The first problem concerns the manner of vector composition, which ignores the relation between the target a and its context b. This relation can have a decisive influence on their interpretation, as Example (2) shows:

(2) a. a horse draws
 b. draw a horse

In (2a), the meaning of the verb *draw* can be paraphrased as *pull*, while in (2b) it is similar to *sketch*. This difference in meaning is due to the difference in relation: in (2a), *horse* is the subject, while in (2b) it is the object. On the modeling side, however, a vector combination function that ignores the relation will assign the same representation to (2a) and (2b). Thus, existing models are systematically unable to capture this class of phenomena.

Single vectors are too weak to represent phrases. The second problem arises in the context of the important open question of how semantic spaces can "scale up" to provide interesting meaning representations for entire sentences. We believe that the current vector composition methods, which result in a single vector c, are not informative enough for this purpose. One proposal for "scaling up" is to straightforwardly interpret $c = a \odot b$ as the *meaning of the phrase* $a + b$ (Kintsch, 2001; Mitchell and Lapata, 2008). The problem is that the vector c can only encode a fixed amount of structural information if its dimensionality is fixed, but there is no upper limit on sentence length, and hence on the amount of structure to be encoded. It is difficult to conceive how c could encode *deeper* semantic properties, like predicate-argument structure (distinguishing "dog bites man" and "man bites dog"), that are crucial for sentence-level semantic tasks such as the recognition of textual entailment (Dagan et al., 2006). An alternative approach to sentence meaning would be to use the vector space representation only for representing word meaning, and to represent sentence structure separately. Unfortunately, present models cannot provide this grounding either, since they compute a single vector c that provides the same representations for *both* the meanings of a and b in context.

In this paper, we propose a new, *structured vector space* model for word meaning (SVS) that addresses these problems. A SVS representation of a lemma comprises several vectors representing the word's lexical meaning as well as the *selectional preferences* that it has for its argument positions. The meaning of word a in context b is computed by combining a with b's selectional preference vector specific to the relation between a and b, addressing the first problem above. In an expression $a + b$, the meanings of a and b in this context are computed as two separate vectors a' and b'. These vectors can then be combined with a representation of the structure's expression (e.g., a parse tree), to address the second problem discussed above. We test the SVS model on the task of recognizing contextually appropriate paraphrases, finding that SVS performs at and above the state-of-the-art.

Plan of the paper. Section 2 reviews related work. Section 3 presents the SVS model for word meaning in context. Sections 4 to 6 relate experiments on the paraphrase appropriateness task.

2 Related Work

In this section we give a short overview over existing vector space based approaches to computing word meaning in context.

General context effects. The first category of models aims at integrating the widest possible range of context information without recourse to linguistic structure. The best-known work in this category is Schütze (1998). He first computes "first-order" vector representations for word meaning by collecting co-occurrence counts from the entire corpus. Then, he determines "second-order" vectors for individual word instances in their context, which is taken to be a simple surface window, by summing up all first-order vectors of the words in this context. The resulting vectors form sense clusters.

McDonald and Brew (2004) present a similar model. They compute the expectation for a word w_i in a sequence by summing the first-order vectors for the words w_1 to w_{i-1} and showed that the distance between expectation and first-order vector for w_i correlates with human reading times.

Predicate-argument combination. The second category of prior studies concentrates on contexts consisting of a single word only, typically modeling the combination of a predicate p and an argument a. Kintsch (2001) uses vector representations of p and a to identify the set of words that are similar to both p and a. After this set has been narrowed down in a self-inhibitory network, the meaning of the predicate-argument combination is obtained by computing the centroid of its members' vectors. The procedure does not take the relation between p and a into account.

Mitchell and Lapata (2008) propose a framework to represent the meaning of the combination $p + a$ as a function f operating on four components:

$$c = f(p, a, R, K) \quad (3)$$

R is the relation holding between p and a, and K additional knowledge. This framework allows sensitivity to the relation. However, the concrete instantiations that Mitchell and Lapata consider disregards K and R, thus sharing the other models' limitations. They focus instead on methods for the direct combination of p and a: In a comparison between component-wise addition and multiplication of p and a, they find far superior results for the multiplication approach.

Tensor product-based models. Smolensky (1990) uses tensor product to combine two word vectors a and b into a vector c representing the expression $a+b$. The vector c is located in a very high-dimensional space and is thus capable of encoding the structure of the expression; however, this makes the model infeasible in practice, as dimensionality rises with every word added to the representation. Jones and Mewhort (2007) represent lemma meaning by using circular convolution to encode n-gram co-occurrence information into vectors of fixed dimensionality. Similar to Brew and McDonald (2004), they predict most likely next words in a sequence, without taking syntax into account.

Kernel methods. One of the main tests for the quality of models of word meaning in context is the ability to predict the appropriateness of paraphrases in given a context. Typically, a paraphrase applies only to some senses of a word, not all, as can be seen in the paraphrases "grab" and "contract" of "catch". Vector space models generally predict paraphrase appropriateness based on the similarity between vectors. This task can also be addressed with kernel methods, which project items into an implicit feature space for efficient similarity computation. Consequently, vector space methods and kernel methods have both been used for NLP tasks based on similarity, notably Information Retrieval and Textual Entailment. Nevertheless, they place their emphasis on different types of information. Current kernels are mostly tree kernels that compare syntactic structure, and use semantic information mostly for smoothing syntactic similarity (Moschitti and Quarteroni, 2008). In contrast, vector-space models focus on the interaction between the lexical meaning of words in composition.

3 A structured vector space model for word meaning in context

In this section, we define the structured vector space (SVS) model of word meaning.

The main intuition behind our model is to view the interpretation of a word in context as guided by *expectations about typical events.* For example, in (1a), we assume that upon hearing the phrase "catch a ball", the hearer will interpret the meaning of "catch" to match *typical actions that can be performed with a ball.* Similarly, the interpretation of "ball" will reflect the hearer's expectations about *typical things that can be caught.* This move to include typical arguments and predicates into a model of word meaning can be motivated both on cognitive and linguistic grounds.

In cognitive science, the central role of expectations about typical events for human language processing is well-established. Expectations affect reading times (McRae et al., 1998), the interpretation of participles (Ferretti et al., 2003), and sentence processing generally (Narayanan and Jurafsky, 2002; Padó et al., 2006). Expectations exist both for verbs and nouns (McRae et al., 1998; McRae et al., 2005).

In linguistics, expectations, in the form of *selectional restrictions* and *selectional preferences*, have long been used in semantic theories (Katz and Fodor, 1964; Wilks, 1975), and more recently induced from corpora (Resnik, 1996; Brockmann and Lapata, 2003). Attention has mostly been limited to selectional preferences of verbs, which have been used for example for syntactic disambiguation (Hindle and Rooth, 1993), word sense disambiguation (McCarthy and Carroll, 2003) and semantic role labeling (Gildea and Jurafsky, 2002). Recently, a vector-spaced model of selectional preferences has been proposed that computes the typicality of an argument simply through similarity to previously seen arguments (Erk, 2007; Padó et al., 2007).

We first present the SVS model of word meaning

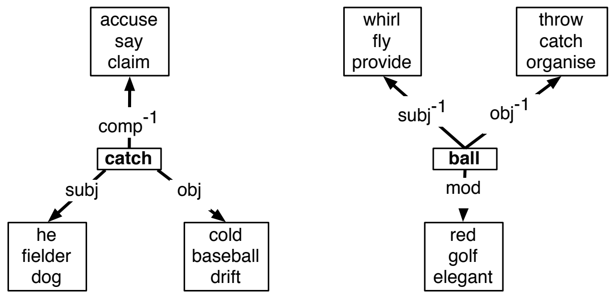


Figure 1: Structured meaning representations for noun *ball* and verb *catch*: lexical information plus expectations

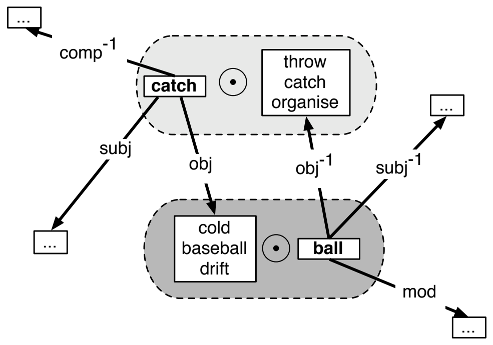


Figure 2: Combining predicate and argument via relation-specific semantic expectations

that integrates lexical information with selectional preferences. Then, we show how the SVS model provides a new way of computing meaning in context.

Representing lemma meaning. We abandon the traditional choice of representing word meaning as a single vector. Instead, we encode each word as a combination of (a) one vector that models the *lexical* meaning of the word, and (b) a set of vectors, each of which represents the *semantic expectations/selectional preferences* for one particular relation that the word supports.[1]

The idea is illustrated in Fig. 1. In the representation of the verb *catch*, the central square stands for the lexical vector of *catch* itself. The three arrows link it to *catch*'s preferences for its subjects ($subj$), its objects (obj), and for verbs for which it appears as a complement ($comp^{-1}$). The figure shows the selectional preferences as word lists for readability; in practice, each selectional preference is a single vector (cf. Section 4). Likewise, *ball* is represented by one vector for *ball* itself, one for *ball*'s preferences for its modifiers (mod), one vector for the verbs of which it is a subject ($subj^{-1}$), and one for the verbs of which is an object (obj^{-1}).

This representation includes selectional preferences (like $subj$, obj, mod) exactly parallel to *inverse* selectional preferences ($subj^{-1}$, obj^{-1}, $comp^{-1}$). To our knowledge, preferences of the latter kind have not been studied in computational linguistics. However, their existence is supported in psycholinguistics by priming effects from nouns to typical verbs (McRae et al., 2005).

Formally, let D be a vector space (the set of possible vectors), and let $\mathcal{R}$ be some set of relation labels. In the *structured vector space (*SVS*)* model, we represent the meaning of a lemma w as a triple

$$w = (v, R, R^{-1})$$

where $v \in D$ is a lexical vector describing the word w itself, $R : \mathcal{R} \to D$ maps each relation label onto a vector that describes w's selectional preferences, and $R^{-1} : \mathcal{R} \to D$ maps from role labels to vectors describing inverse selectional preferences of w. Both R and R^{-1} are partial functions. For example, the direct object preference would be undefined for intransitive verbs.

Computing meaning in context. The SVS model of lemma meaning permits us to compute the meaning of a word a in the context of another word b in a new way, via their selectional preferences. Let (v_a, R_a, R_a^{-1}) and (v_b, R_b, R_b^{-1}) be the representations of the two words, and let $r \in \mathcal{R}$ be the relation linking a to b. Then, we define the meaning of a and b in this context as a pair (a', b') of vectors, where a' is the meaning of a in the context of b, and b' the meaning of b in the context of a:

$$\begin{aligned} a' &= \left(v_a \odot R_b^{-1}(r), R_a - \{r\}, R_a^{-1}\right) \\ b' &= \left(v_b \odot R_a(r), R_b, R_b^{-1} - \{r\}\right) \end{aligned} \quad (4)$$

where $v_1 \odot v_2$ is a direct vector combination function as in traditional models, e.g. addition or component-wise multiplication. If either $R_a(r)$ or $R_b^{-1}(r)$ are not defined, the combination fails. Afterwards, the argument position r is considered filled, and is deleted from R_a and R_b^{-1}.

[1]We do not commit to a particular set of relations; see the discussion at the end of this section.

Figure 2 illustrates this procedure on the representations from Figure 1. The dotted lines indicate that the lexical vector for *catch* is combined with the inverse object preference of *ball*. Likewise, the lexical vector for *ball* is combined with the object preference vector of *catch*.

Note that our procedure for computing meaning in context can be expressed within the framework of Mitchell and Lapata (Eq. (3)). We can encode the expectations of a and b as additional knowledge K. The combined representation c is the pair (a', b') that is computed according to our model (Eq. (4)).

The SVS scheme we have proposed incorporates syntactic information in a more general manner than previous models, and thus addresses the issues we have discussed in Section 1. Since the representation retains individual selectional preferences for all relations, combining the same words through different relations can (and will in general) result in different adapted representations. For instance, in the case of Example (2), we would expect the inverse subject preference of *horse* ("things that a horse typically does") to push the lexical vector of *draw* into the direction of *pulling*, while its inverse object preference ("things that are done to horses") suggest a different interpretation.

Rather than yielding a single, joint vector for the whole expression, our procedure for computing meaning in context results in one context-adapted meaning representation per word, similar to the output of a WSD system. As a consequence, our model can be combined with any formalism representing the structure of an expression. (The formalism used then determines the set $\mathcal{R}$ of relations.) For example, combining SVS with a dependency tree would yield a tree in which each node is labeled by a SVS tuple that represents the word's meaning in context.

4 Experimental setup

This section provides the background to the following experimental evaluation of SVS, including parameters used for computing the SVS representations that will be used in the experiments.

4.1 Experimental rationale

In this paper, we evaluate the SVS model against the task of predicting, given a predicate-argument pair, how appropriate a paraphrase (of either the predicate or the argument) is in that context. We perform two experiments that both use the paraphrase task, but differ in their emphasis. Experiment 1 replicates an existing evaluation against human judgments. This evaluation uses synthetic dataset, limited to one particular construction, and constructed to provide maximally distinct paraphrase candidates. Experiment 2 considers a broader class of constructions along with annotator-generated paraphrase candidates that are not screened for distinctness. In both experiments, we compare the SVS model against the state-of-the-art model by Mitchell and Lapata 2008 (henceforth M&L; cf. Sec. 2 for model details).

4.2 Parameter choices

Vector space. In our parameterization of the vector space, we largely follow M&L because their model has been rigorously evaluated and found to outperform a range of other models.

Our first space is a traditional "bag-of-words" vector space (BOW, (Lund and Burgess, 1996)). For each pair of a target word and context word, the BOW space records a function of their co-occurrence frequency within a surface window of size 10. The space is constructed from the British National Corpus (BNC), and uses the 2,000 most frequent context words as dimensions.

We also consider a "dependency-based" vector space (SYN, (Padó and Lapata, 2007)). In this space, target and context words have to be linked by a "valid" dependency path in a dependency graph to count as co-occurring.[2] This space was built from BNC dependency parses obtained from Minipar (Lin, 1993).

For both spaces, we used pre-experiments to compare two methods for the computation of vector components, namely raw co-occurrence counts, the standard model, and the pointwise mutual information (PMI) definition employed by M&L.

Selectional preferences. We use a simple, knowledge-lean representation for selectional preferences inspired by Erk (2007), who models selectional preference through similarity to seen filler vectors $\vec{v}_a$: We compute the selectional preference vector for word b and relation r as the weighted

[2]More specifically, we used the minimal context specification and plain weight function. See Padó and Lapata (2007).

centroid of seen filler vectors $\vec{v}_a$. We collect seen fillers from the Minipar-parse of the BNC.

Let $f(a, r, b)$ denote the frequency of a occurring in relation r to b in the parsed BNC, then

$$R_b(r)_{\text{SELPREF}} = \sum_{a:f(a,r,b)>0} f(a, r, b) \cdot \vec{v}_a \quad (5)$$

We call this base model SELPREF. We will also study two variants of SELPREF, based on two different hypotheses about what properties of the selectional preferences are particularly important for meaning adaption. The first model aims specifically at alleviating noise introduced by infrequent fillers, a common problem in data-driven approaches. It only uses fillers seen more often than a threshold θ. We call this model SELPREF-CUT:

$$R_b(r)_{\text{SELPREF-CUT}} = \sum_{a:f(a,r,b)>\theta} f(a, r, b) \cdot \vec{v}_a \quad (6)$$

Our second variant again aims at alleviating noise, but noise introduced by low-valued dimensions rather than infrequent fillers. It achieves this by taking each component of the selectional preference vector to the nth power. In this manner, dimensions with high counts are further inflated, while dimensions with low counts are depressed.[3] This model, SELPREF-POW, is defined as follows: If $R_b(r)_{\text{SELPREF}} = \langle v_1, \ldots, v_m \rangle$,

$$R_b(r)_{\text{SELPREF-POW}} = \langle v_1^n, \ldots, v_m^n \rangle \quad (7)$$

The inverse selectional preferences R_b^{-1} are defined analogously for all three model variants. We instantiate the vector combination function $\odot$ as component-wise multiplication, following M&L.

Baselines and significance testing. All tasks that we consider below involve judgments for the meaning of a word a in the context of a word b. A first baseline that every model must beat is simply using the original vector for a. We call this baseline "target only". Since we assume that the selectional preferences of b model the expectations for a, we use b's selectional preference vector for the given relation as a second baseline, "selpref only".

[3]Since we focus on the size-invariant cosine similarity, the use of this model does not require normalization.

verb	subject	landmark	sim	judgment
slump	shoulder	slouch	high	7
slump	shoulder	decline	low	2
slump	value	slouch	low	3
slump	value	decline	high	7

Figure 3: Experiment 1: Human similarity judgements for subject-verb pair with high- and low-similarity landmarks

Differences between the performance of models were tested for significance using a stratified shuffling-based randomization test (Yeh, 2000).[4].

5 Exp. 1: Predicting similarity ratings

In our first experiment, we attempt to predict human similarity judgments. This experiment is a replication of the evaluation of M&L on their dataset[5].

Dataset. The M&L dataset comprises a total of 3,600 human similarity judgements for 120 experimental items. Each item, as shown in Figure 3, consists of an intransitive verb and a subject noun that are combined with a "landmark", a synonym of the verb that is chosen to be either similar or dissimilar to the verb in the context of the given subject.

The dataset was constructed by extracting pairs of subjects and intransitive verbs from a parsed version of the BNC. Each item was paired with two landmarks, chosen to be as dissimilar as possible according to a WordNet similarity measure. All nouns and verbs were subjected to a pretest, where only those with highly significant variations in human judgments across landmarks were retained.

For each item of the final dataset, judgements on a 7-point scale were elicited. For example, judges considered the compatible landmark "slouch" to be much more similar to "shoulder slumps" than the incompatible landmark "decline". In Figure 3, the column *sim* shows whether the experiment designers considered the respective landmark to have high or low similarity to the verb, and the column *judgment* shows a participant's judgments.

Experimental procedure. We used cosine to compute similarity to the lexical vector of the landmark.

[4]The software is available at `http://www.nlpado.de/~sebastian/sigf.html`.

[5]We thank J. Mitchell and M. Lapata for providing their data.

Model	high	low	ρ
BOW space			
Target only	0.32	0.32	0.0
Selpref only	0.46	0.4	0.06**
M&L	0.25	0.15	0.20**
SELPREF	0.32	0.26	0.12**
SELPREF-CUT, θ=10	0.31	0.24	0.11**
SELPREF-POW, n=20	0.11	0.03	**0.27****
Upper bound	–	–	0.4
SYN space			
Target only	0.2	0.2	0.08**
Selpref only	0.27	0.21	0.16**
M&L	0.13	0.06	**0.24****
SELPREF	0.22	0.16	0.13**
SELPREF-CUT, θ=10	0.2	0.13	0.13**
SELPREF-POW, n=30	0.08	0.04	0.22**
Upper bound	–	–	0.4

Table 1: Experiment 1: Mean cosine similarity for items with high- and low-similarity landmarks; correlation with human judgements (ρ). (**: p $<$ 0.01)

"Target only" compares the landmark against the lexical vector of the verb, and "selpref only" compares it to the noun's $subj^{-1}$ preference. For the M&L model, the comparison is to the combined lexical vectors of verb and noun. For our models SELPREF, SELPREF-CUT and SELPREF-POW, we combine the verb's lexical vector with the $subj^{-1}$ preference of the noun. We used a held-out dataset of 10% of the data to optimize the parameters of θ of SELPREF-CUT and n of SELPREF-POW. Vectors with PMI components could model the data, while raw frequency components could not; we report only the former.

We use the same two evaluation scores as M&L: The first score is the average similarity to compatible landmarks (high) and incompatible landmarks (low). The second is Spearman's ρ, a nonparametric correlation coefficient. We compute ρ between individual human similarity scores and our predictions. Based on agreement between human judges, M&L estimate an upper bound ρ of 0.4 for the dataset.

Results and discussion. Table 1 shows the results of Exp. 1 on the test set. In the upper half (BOW), we replicate M&L's main finding that simple component-wise multiplication of the predicate and argument vectors results in a highly significant correlation of $\rho = 0.2$, significantly outperforming both baselines. It is interesting, though, that the $subj^{-1}$ preference itself ("Selpref only") is already highly significantly correlated with the human judgments.

A comparison of the upper half (BOW) with the lower half (SYN) shows that the dependency-based space generally shows better correlation with human judgements. This corresponds to a beneficial effect of syntactic information found for other applications of semantic spaces (Lin, 1998; Padó and Lapata, 2007).

All instances of the SELPREF model show highly significant correlations. SELPREF and SELPREF-CUT show very similar performance. They do better than both baselines in the BOW space; however, in the cleaner SYN space, their performance is numerically lower than using selectional preferences only ($\rho = 0.13$ vs. 0.16). SELPREF-POW is always significantly better than SELPREF and SELPREF-CUT, and shows the best result of all tested models ($\rho = 0.27$, BOW space). The performance is somewhat lower in the SYN space ($\rho = 0.22$). However, this difference, and the difference to the best M&L model at $\rho = 0.24$, are not statistically significant.

The SVS model computes meaning in context by combining a word's lexical representation with the preference vector of its context. In this, it differs from previous models, including that by M&L, which used what we have been calling "direct combination". So it is important to ask to what extent this difference in method translate to a difference in predictions. We analyzed this by measuring the similarity by the nouns' lexical vectors, used by direct combination methods, and their inverse subject preferences, which SVS uses. The result is shown in the first column in Table 2, computed as mean cosine similarities and standard deviations between noun vectors and selectional preferences. The table shows that these vectors have generally low similarity, which is further

Model	lex. vector	obj^{-1} selpref
SELPREF	0.23 (0.09)	0.88 (0.07)
SELPREF-CUT (10)	0.20 (0.10)	0.72 (0.18)
SELPREF-POW (30)	0.03 (0.08)	0.52 (0.48)

Table 2: Experiment 1: Average similarity (and standard deviation) between the inverse subject preferences of a noun and (left) its lexical vector and (right) inverse object preferences vector (cosine similarity in SYN space)

reduced by applying cutoff and potentiation. Thus, the predictions of SVS will differ from those of direct combination models like M&L.

A related question is whether syntax-aware vector combination makes a difference: Does the model encode different expectations for different syntactic relations (cf. Example 2)? The second column of Table 2 explores this question by comparing inverse selectional preferences for the subject and object slots. We observe that the similarity is very high for raw preferences, but becomes lower when noise is eliminated. Since the SELPREF-POW model performed best in our evaluation, we read this as evidence that potentiation helps to suppress noise introduced by mis-identified subject and object fillers.

In Experiment 1, all experimental items were verbs, which means that all disambiguation was done through inverse selectional preferences. As inverse selectional preferences are currently largely unexplored, it is interesting to note that the evidence that they provide for the paraphrase task is as strong as that of the context nouns themselves.

6 Exp. 2: Ranking paraphrases

This section reports on a second, more NLP-oriented experiment whose task is to distinguish between appropriate and inappropriate paraphrases on a broader range of constructions.

Dataset. For this experiment, we use the SemEval-1 *lexical substitution* (lexsub) dataset (McCarthy and Navigli, 2007), which contains 10 instances each of 200 target words in sentential contexts, drawn from Sharoff's (2006) English Internet Corpus. Contextually appropriate paraphrases for each instance of each target word were elicited from up to 6 participants. Fig. 4 shows two instances for the verb *to work*. The distribution over paraphrases can be seen as a characterization of the target word's meaning in each context.

Experimental procedure. In this paper, we predict appropriate paraphrases solely on the basis of a single context word that stands in a direct predicate-argument relation to the target word. We extracted all instances from the lexsub test data with such a relation. After parsing all sentences with verbal and nominal targets with Minipar, this resulted in three

Sentence	Substitutes
By asking people who **work** there, I have since determined that he didn't. (# 2002)	be employed 4; labour 1
Remember how hard your ancestors **worked**. (# 2005)	toil 4; labour 3; task 1

Figure 4: Lexical substitution example items for "work"

sets of sentences: (a), target intransitive verbs with noun subjects (V-SUBJ, 48 sentences); (b), target transitive verbs with noun objects (V-OBJ, 213 sent.); and (c), target nouns occurring as objects of verbs (N-OBJ, 102 sent.).[6] Note that since we use only part of the lexical substitution dataset in this experiment, a direct comparison with results from the SemEval task is not possible.

As in the original SemEval task, we phrase the task as a ranking problem. For each target word, the paraphrases given for all 10 instances are pooled. The task is to rank the list for each item so that appropriate paraphrases (such as "be employed" for # 2002) rank higher than paraphrases not given (e.g., "toil").

Our model ranks paraphrases by their similarity to the following combinations (Eq. (4)): for V-SUBJ, verb plus the noun's $subj^{-1}$ preferences; for V-OBJ, verb plus the noun's obj^{-1} preferences; and for N-OBJ, the noun plus the verb's obj preferences. Our comparison model, M&L, ranks all paraphrases by their similarity to the direct noun-verb combination.

To avoid overfitting, we consider only the two models that performed optimally in in the SYN space in Experiment 1 (SELPREF-POW with n=30 and M&L). However, since we found that vectors with raw frequency components could model the data, while PMI components could not, we only report the former.

For evaluation, we adopt the SemEval "out of ten" precision metric $\mathrm{P_{OOT}}$. It uses the model's ten top-ranked paraphrases as its guesses for appropriate paraphrases. Let G_i be the gold paraphrases for item i, M_i the model's top ten paraphrases for i, and $f(s,i)$ the frequency of s as paraphrase for i:

$$\mathrm{P_{OOT}} = 1/|I| \sum_i \frac{\sum_{s \in M_i \cap G_i} f(s,i)}{\sum_{s \in G_i} f(s,i)} \qquad (8)$$

McCarthy and Navigli propose this metric for the

[6]The specification of this dataset will be made available.

Model	V-SUBJ	V-OBJ	N-OBJ
Target only	47.9	47.4	49.6
Selpref only	54.8	51.4	55.0
M&L	50.3	52.0	53.4
SELPREF-POW, n=30	**63.1**	**55.8**	**56.9**

Table 3: Experiment 2: Mean "out of ten" precision (P_{OOT})

dataset for robustness. Due to the sparsity of paraphrases, a metric that considers fewer guesses leads to artificially low results when a "good" paraphrase was not mentioned by the annotators by chance but is ranked highly by a model.

Results and discussion. Table 6 shows the mean out-of-ten precision for all models. The behavior is fairly uniform across all three datasets. Unsurprisingly, "target only", which uses the same ranking for all instances of a target, yields the worst results.[7]

M&L's direct combination model outperforms "target only" significantly ($p < 0.05$). However, on both the V-SUBJ and the N-OBJ the "selpref only" baseline does better than direct combination. The best results on all datasets are obtained by SELPREF-POW. The difference between SELPREF-POW and the "target only" baseline is highly significant ($p < 0.01$). The difference to M&L's model is significant at $p = 0.05$.

We interpret these results as encouraging evidence for the usefulness of selectional preferences for judging substitutability in context. Knowledge about the selectional preferences of a single context word can already lead to a significant improvement in precision. We find this overall effect even though the word is not informative in all cases. For instance, the subject of item 2002 in Fig. 4, "who", presumably helps little in determining the verb's context-adapted meaning.

It is interesting that the improvement of SELPREF-POW over "selpref only" is smallest for the N-OBJ dataset (1.9% P_{OOT}). N-OBJ uses selectional preferences for nouns that may fill the direct object position, , while V-SUBJ and V-OBJ use inverse selectional preferences for verbs (cf. the two graphs in Fig. 1).

[7]"Target only" still does very much better than a random baseline, which performs at 22% P_{OOT}.

7 Conclusion

In this paper, we have considered semantic space models that can account for the meaning of word occurrences in context. Arguing that existing models do not sufficiently take syntax into account, we have introduced the new structured vector space (SVS) model of word meaning. In addition to a vector representing a word's lexical meaning, it contains vectors representing the word's selectional preferences. These selectional preferences play a central role in the computation of meaning in context.

We have evaluated the SVS model on two datasets on the task of predicting the felicitousness of paraphrases in given contexts. On the M&L dataset, SVS outperforms the state-of-the-art model of M&L, though the difference is not significant. On the Lexical Substitution dataset, SVS significantly outperforms the state-of-the-art. This is especially interesting as the Lexical Substitution dataset, in contrast to the M&L data, uses "realistic" paraphrase candidates that are not necessarily maximally distinct.

The most important limitation of the evaluation that we have given in this paper is that we have only considered single words as context. Our next step will be to integrate information from multiple relations (such as both the subject and object positions of a verb) into the computation of context-specific meaning. Our eventual aim is a model that can give a compositional account of a word's meaning in context, where all words in an expression disambiguate one another according to the relations between them.

We will explore the usability of vector space models of word meaning in NLP applications, formulated as the question of how to perform inferences on them in the context of the Textual Entailment task (Dagan et al., 2006). Paraphrase-based inference rules play a large role in several recent approaches to Textual Entailment (e.g. Szpektor et al (2008)); appropriateness judgments of paraphrases in context, the task of Experiments 1 and 2 above, can be viewed as testing the applicability of these inferences rules.

Acknowledgments. Many thanks for helpful discussion to Jason Baldridge, David Beaver, Dedre Gentner, James Hampton, Dan Jurafsky, Alexander Koller, Brad Love, and Ray Mooney.

References

C. Brockmann, M. Lapata. 2003. Evaluating and combining approaches to selectional preference acquisition. In *Proceedings of EACL*, 27–34.

I. Dagan, O. Glickman, B. Magnini. 2006. The PASCAL Recognising Textual Entailment Challenge. In *Machine Learning Challenges*, Lecture Notes in Computer Science, 177–190. Springer.

K. Erk. 2007. A simple, similarity-based model for selectional preferences. In *Proceedings of ACL*, 216–223.

T. Ferretti, C. Gagné, K. McRae. 2003. Thematic role focusing by participle inflections: evidence form conceptual combination. *Journal of Experimental Psychology*, 29(1):118–127.

D. Gildea, D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

D. Hindle, M. Rooth. 1993. Structural ambiguity and lexical relations. *Computational Linguistics*, 19(1):103–120.

M. Jones, D. Mewhort. 2007. Representing word meaning and order information in a composite holographic lexicon. *Psychological review*, 114:1–37.

J. J. Katz, J. A. Fodor. 1964. The structure of a semantic theory. In *The Structure of Language*. Prentice-Hall.

A. Kilgarriff. 1997. I don't believe in word senses. *Computers and the Humanities*, 31(2):91–113.

W. Kintsch. 2001. Predication. *Cognitive Science*, 25:173–202.

T. Landauer, S. Dumais. 1997. A solution to Platos problem: the latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240.

D. Lin. 1993. Principle-based parsing without overgeneration. In *Proceedings of ACL*, 112–120.

D. Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING-ACL*, 768–774.

K. Lund, C. Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, and Computers*, 28:203—208.

C. D. Manning, P. Raghavan, H. Schütze. 2008. *Introduction to Information Retrieval*. CUP.

D. McCarthy, J. Carroll. 2003. Disambiguating nouns, verbs, and adjectives using automatically acquired selectional preferences. *Computational Linguistics*, 29(4):639–654.

D. McCarthy, R. Navigli. 2007. SemEval-2007 Task 10: English Lexical Substitution Task. In *Proceedings of SemEval*, 48–53.

S. McDonald, C. Brew. 2004. A distributional model of semantic context effects in lexical processing. In *Proceedings of ACL*, 17–24.

S. McDonald, M. Ramscar. 2001. Testing the distributional hypothesis: The influence of context on judgements of semantic similarity. In *Proceedings of CogSci*, 611–616.

K. McRae, M. Spivey-Knowlton, M. Tanenhaus. 1998. Modeling the influence of thematic fit (and other constraints) in on-line sentence comprehension. *Journal of Memory and Language*, 38:283–312.

K. McRae, M. Hare, J. Elman, T. Ferretti. 2005. A basis for generating expectancies for verbs from nouns. *Memory and Cognition*, 33(7):1174–1184.

J. Mitchell, M. Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL*, 236–244.

A. Moschitti, S. Quarteroni. 2008. Kernels on linguistic structures for answer extraction. In *Proceedings of ACL*, 113–116, Columbus, OH.

S. Narayanan, D. Jurafsky. 2002. A Bayesian model predicts human parse preference and reading time in sentence processing. In *Proceedings of NIPS*, 59–65.

S. Padó, M. Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

U. Padó, F. Keller, M. W. Crocker. 2006. Combining syntax and thematic fit in a probabilistic model of sentence processing. In *Proceedings of CogSci*, 657–662.

S. Padó, U. Padó, K. Erk. 2007. Flexible, corpus-based modelling of human plausibility judgements. In *Proceedings of EMNLP/CoNLL*, 400–409.

P. Resnik. 1996. Selectional constraints: An information-theoretic model and its computational realization. *Cognition*, 61:127–159.

G. Salton, A. Wang, C. Yang. 1975. A vector-space model for information retrieval. *Journal of the American Society for Information Science*, 18:613–620.

H. Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–124.

S. Sharoff. 2006. Open-source corpora: Using the net to fish for linguistic data. *International Journal of Corpus Linguistics*, 11(4):435–462.

P. Smolensky. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46:159–216.

I. Szpektor, I. Dagan, R. Bar-Haim, J. Goldberger. 2008. Contextual preferences. In *Proceedings of ACL*, 683–691, Columbus, OH.

Y. Wilks. 1975. Preference semantics. In *Formal Semantics of Natural Language*. CUP.

A. Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceeedings of COLING*, 947–953.

Learning Graph Walk Based Similarity Measures for Parsed Text

Einat Minkov[*]
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
einat@cs.cmu.edu

William W. Cohen
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
wcohen@cs.cmu.edu

Abstract

We consider a parsed text corpus as an instance of a labelled directed graph, where nodes represent words and weighted directed edges represent the syntactic relations between them. We show that graph walks, combined with existing techniques of supervised learning, can be used to derive a task-specific word similarity measure in this graph. We also propose a new *path-constrained* graph walk method, in which the graph walk process is guided by high-level knowledge about meaningful edge sequences (paths). Empirical evaluation on the task of named entity coordinate term extraction shows that this framework is preferable to vector-based models for small-sized corpora. It is also shown that the path-constrained graph walk algorithm yields both performance and scalability gains.

1 Introduction

Graph-based similarity measures have been used for a variety of language processing applications. In this paper we assume directed graphs, where typed nodes denote entities and labelled directed and weighted edges denote the relations between them. In this framework, graph walks can be applied to draw a measure of similarity between the graph nodes. Previous works have applied graph walks to draw a notion of semantic similarity over such graphs that were carefully designed and manually tuned, based on WordNet reations (Toutanova et al., 2004; Collins-Thompson and Callan, 2005; Hughes and Ramage, 2007).

While these and other researchers have used WordNet to evaluate similarity between words, there has been much interest in extracting such a measure from text corpora (e.g., (Snow et al., 2005; Padó and Lapata, 2007)). In this paper, we suggest processing dependency parse trees within the general framework of directed labelled graphs. We construct a graph that directly represents a corpus of structured (parsed) text. In the suggested graph scheme, nodes denote words and weighted edges represent the dependency relations between them. We apply graph walks to derive an inter-word similarity measure. We further apply learning techniques, adapted to this framework, to improve the derived corpus-based similarity measure.

The learning methods applied include existing learning techniques, namely edge weight tuning, where weights are associated with the edge types, and discriminative reranking of graph nodes, using features that describe the possible paths between a graph node and the initial "query nodes" (Minkov and Cohen, 2007).

In addition, we outline in this paper a novel method for learning *path-constrained* graph walks. While reranking allows use of high-level features that describe properties of the traversed paths, the suggested algorithm incorporates this information in the graph walk process. More specifically, we allow the probability flow in the graph to be conditioned on the history of the walk. We show that this method results in improved performance as it directs probability flow to meaningful paths. In addition, it leads

[*]Current address: Nokia Research Center Cambridge, Cambridge, MA 02142, USA.

to substantial gains in terms of runtime performance.

The graph representation and the set of learning techniques suggested are empirically evaluated on the task of *coordinate term* extraction[1] from small to moderately sized corpora, where we compare them against vector-based models, including a state-of-the-art syntactic distributional similarity method (Padó and Lapata, 2007). It is shown that the graph walk based approach gives preferable results for the smaller datasets (and comparable otherwise), where learning yields significant gains in accuracy.

There are several contributions of this paper. First, we represent dependency-parsed corpora within a general graph walk framework, and derive inter-word similarity measures using graph walks and learning techniques available in this framework. To our knowledge, the application of graph walks to parsed text in general, and to the extraction of coordinate terms in particular, is novel. Another main contribution of this paper is the path-constrained graph walk variant, which is a general learning technique for calculating the similarity between graph nodes in directed and labelled graphs.

Below we first outline our proposed scheme for representing a dependency-parsed text corpus as a graph, and provide some intuitions about the associated similarity metric (Section 2). We then give an overview of the graph-walk based similarity metric (Section 3), as well as the known edge weight tuning and reranking learning techniques (Section 4). We next present the proposed algorithm of path-constrained graph walks (Section 5). The paper proceeds with a review of related work (Section 6), a discussion of the coordinate term extraction task, empirical evaluation and our conclusions (Sections 7-9).

2 Representing a Corpus as a Graph

A typed dependency parse tree consists of directed links between words, where dependencies are labelled with the relevant grammatical relation (e.g., *nominal subject*, *indirect object* etc.). We suggest representing a text corpus as a connected graph of dependency structures, according to the scheme shown in Figure 1. The graph shown in the figure

[1]In particular, we focus on the extraction of named entity classes.

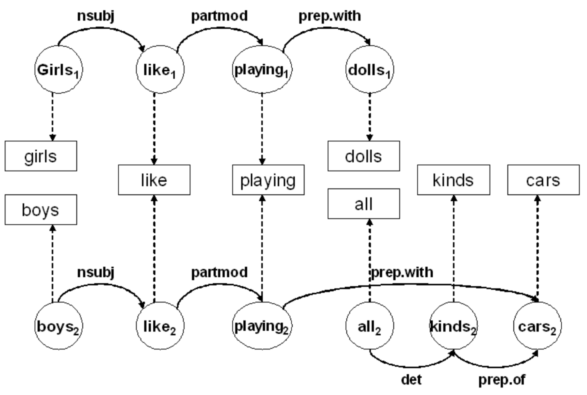


Figure 1: The suggested graph schema, demonstrated for a two-sentence corpus.

includes the dependency analysis of two sentences: “boys like playing with all kinds of cars”, and “girls like playing with dolls”. In the graph, each word mention is represented as a node, which includes the index of the sentence in which it appears, as well as its position within the sentence. Word mentions are marked as circles in the figure. The “type” of each word – henceforth a *term* node – is denoted by a square in the figure. Each word mention is linked to the corresponding term; for example, the nodes “like$_1$” and “like$_2$” represent distinct word mentions and both nodes are linked to the *term* “like”. For every edge in the graph, we add another edge in the opposite direction (not shown in the figure); for example, an inverse edge exists from “like$_1$” to “girls$_1$” with an edge labelled as “nsubj-inv”. The resulting graph is highly interconnected and cyclic.

We will apply graph walks to derive an extended measure of similarity, or relatedness, between word *terms* (as defined above). For example, starting from the term “girls”, we will reach the semantically related term “boys” via the following two paths:

(1) girls $\xrightarrow{mention}$ girls1 $\xrightarrow{nsubj}$ like1 $\xrightarrow{as-term}$ like $\xrightarrow{mention}$ like2 $\xrightarrow{nsubj-inverse}$ boys2 $\xrightarrow{as-term}$ boys

(2) girls $\xrightarrow{mention}$ girls1 $\xrightarrow{nsubj}$ like1 $\xrightarrow{partmod}$ playing1 $\xrightarrow{as-term}$ playing $\xrightarrow{mention}$ playing2 $\xrightarrow{partmod-inverse}$ like2 $\xrightarrow{nsubj-inverse}$ boys2 $\xrightarrow{as-term}$ boys .

Intuitively, in a graph representing a large corpus, terms that are more semantically related will be linked by a larger number of connecting paths. In addition, shorter connecting paths may be in general more meaningful. In the next section we show that the graph walk paradigm addresses both of these requirements. Further, different edge types, as well as the paths traversed, are expected to have varying importance in different types of word similarity (for example, verbs and nouns are associated with different connectivity patterns). These issues are addressed using learning.

3 Graph Walks and Similarity Queries

This section provides a quick overview of the graph walk induced similarity measure. For details, the reader is referred to previous publications (e.g., (Toutanova et al., 2004; Minkov and Cohen, 2007)).

In summary, similarity between two nodes in the graph is defined by a weighted graph walk process, where an edge of type ℓ is assigned an edge weight, θ_ℓ, determined by its type.[2] The transition probability of reaching node y from node x over a single time step, $Pr(x \longrightarrow y)$, is defined as the weight of their connecting edge, θ_l, normalized by the total outgoing weight from x. Given these transition probabilities, and starting from an initial distribution V_q of interest (a *query*), we perform a graph walk for a finite number of steps K. Further, at each step of the walk, a proportion γ of the probability mass at every node is emitted. Thus, this model applies exponential decay on path length. The final probability distribution of this walk over the graph nodes, which we denote as R, is computed as follows: $R = \sum_{i=1}^{K} \gamma^i V_q \mathbf{M}^i$, where M is the transition matrix.[3] The answer to a query, V_q, is a list of nodes, ranked by the scores in the final distribution R. In this multi-step walk, nodes that are reached from the query nodes by many shorter paths will be assigned a higher score than nodes connected over fewer longer paths.

4 Learning

We consider a supervised setting, where we are given a dataset of example queries and labels over the graph nodes, indicating which nodes are relevant to which query. For completeness, we describe here two methods previously described by Minkov and Cohen (Minkov and Cohen, 2007): a hill-climbing method that tunes the graph weights; and a reranking method. We also specify the feature set to be used by the reranking method in the domain of parsed text.

4.1 Weight Tuning

There are several motivations for learning the graph weights Θ in this domain. First, some dependency relations – foremost, *subject* and *object* – are in general more salient than others (Lin, 1998; Padó and Lapata, 2007). In addition, dependency relations may have varying importance per different notions of word similarity (e.g., noun vs. verb similarity (Resnik and Diab, 2000)). Weight tuning allows the adaption of edge weights to each *task* (i.e., distribution of queries).

The weight tuning method implemented in this work is based on an error backpropagation hill climbing algorithm (Diligenti et al., 2005). The algorithm minimizes the following cost function:

$$E = \frac{1}{N} \sum_{z \in N} e_z = \frac{1}{N} \sum_{z \in N} \frac{1}{2} (p_z - p_z^{Opt})^2$$

where e_z is the error for a target node z defined as the squared difference between the final score assigned to z by the graph walk, p_z, and some ideal score according to the example's labels, p_z^{Opt}.[4] Specifically, p_z^{Opt} is set to 1 in case that the node z is relevant or 0 otherwise. The error is averaged over a set of example instantiations of size N. The cost function is minimized by gradient descent where the derivative of the error with respect to an edge weight θ_ℓ is derived by decomposing the walk into single time steps, and considering the contribution of each node traversed to the final node score.

4.2 Node Reranking

Reranking of the top candidates in a ranked list has been successfully applied to multiple NLP tasks (Collins, 2002; Collins and Koo, 2005). In essence, discriminative reranking allows the re-ordering of results obtained by methods that perform some form of local search, using features that encode higher level information.

[2]In this paper, we consider either uniform edge weights; or, learn the set of weights Θ from examples.

[3]We tune K empirically and set $\gamma = 0.5$, as in (Minkov and Cohen, 2007).

[4]For every example query, a handful of the retrieved nodes are considered, including both relevant and irrelevant nodes.

A number of features describing the set of paths from V_q can be conveniently computed in the process of executing the graph walk, and it has been shown that reranking using these features can improve results significantly. It has also been shown that reranking is complementary to weight tuning (Minkov and Cohen, 2007), in the sense that the two techniques can be usefully combined by tuning weights, and then reranking the results.

In the reranking approach, for every training example i ($1 \leq i \leq N$), the reranking algorithm is provided with the corresponding output ranked list of l_i nodes. Let z_{ij} be the output node ranked at rank j in l_i, and let $p_{z_{ij}}$ be the probability assigned to z_{ij} by the graph walk. Each output node z_{ij} is represented through m features, which are computed by pre-defined feature functions $f_1, \ldots, f_m$. The *ranking function* for node z_{ij} is defined as:

$$F(z_{ij}, \bar{\alpha}) = \alpha_0 log(p_{z_{ij}}) + \sum_{k=1}^{m} \alpha_k f_k(z_{ij})$$

where $\bar{\alpha}$ is a vector of real-valued parameters. Given a new test example, the output of the model is the output node list reranked by $F(z_{ij}, \bar{\alpha})$. To learn the parameter weights $\bar{\alpha}$, we here applied a boosting method (Collins and Koo, 2005) (see also (Minkov et al., 2006)).

4.2.1 Features

We evaluate the following feature templates. *Edge label sequence* features indicate whether a particular sequence of edge labels ℓ_i occurred, in a particular order, within the set of paths leading to the target node z_{ij}. *Lexical unigram* feature indicate whether a word mention whose lexical value is t_k was traversed in the set of paths leading to z_{ij}. Finally, the *Source-count* feature indicates the number of different source query nodes that z_{ij} was reached from. The intuition behind this last feature is that nodes linked to multiple query nodes, where applicable, are more relevant. For example, for the query term "girl" in the graph depicted in Figure 1, the target node "boys" is described by the features (denoted as *feature-name.feature-value*): *sequence.nsubj.nsubj-inv* (where *mention* and *as-term* edges are omitted) , *lexical.'"like"* etc.

In this work, the features encoded are binary. However, features can be assugned numeric weights that corresponds to the probability of the indicator being true for any path between x and z_{ij} (Cohen and Minkov, 2006).

5 Path-Constrained Graph Walk

While node reranking allows the incorporation of high-level features that describe the traversed paths, it is desirable to incorporate such information earlier in the graph walk process. In this paper, we suggest a variant of a graph-walk, which is *constrained* by path information. Assume that preliminary knowledge is available that indicates the probability of reaching a relevant node after following a particular edge type sequence (path) from the query distribution V_q to some node x. Rather than fix the edge weights Θ, we can evaluate the weights of the outgoing edges from node x dynamically, given the history of the walk (the path) up to this node. This should result in gains in accuracy, as paths that lead mostly to irrelevant nodes can be eliminated in the graph walk process. In addition, scalability gains are expected, for the same reason.

We suggest a path-constrained graph walk algorithm, where path information is maintained in a compact path-tree structure constructed based on training examples. Each vertex in the path tree denotes a particular walk history. In applying the graph walk, the nodes traversed are represented as a set of node pairs, comprised of the graph node and the corresponding vertices in the path tree. The outgoing edge weights from each node pair will be estimated according to the respective vertex in the path tree. This approach needs to address two subtasks: learning of the path-tree; and updating of the graph walk paradigm to co-sample from the graph and the path tree. We next describe these two components in detail.

The Path-Tree

We construct a path-tree T using a training set of N example queries. Let a *path* p be a sequence of $k < K$ edge types (where K is the maximum number of graph walk steps). For each training example, we recover all of the connecting paths leading to the top M (correct and incorrect) nodes. We consider only acyclic paths. Let each path p be associated with its count, within the paths leading to the correct nodes, denoted as C_p^+. Similarly, the

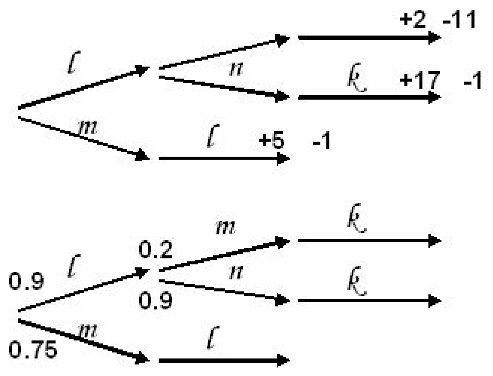


Figure 2: An example path-tree.

count within paths leading to the negatively labelled nodes is denoted C_p^-. The full set of paths observed is then represented as a tree.[5] The leaves of the tree are assigned a Laplace-smoothed probability: $Pr(p) = \frac{C_p^+ + 1}{C_p^+ + C_p^- + 2}$.

Given path probabilities, they are propagated backwards to all tree vertices, applying the MAX operator.[6] Consider the example given in Figure 2. The path-tree in the figure includes three paths (constructed from edge types k, l, m, n). The top part of the figure gives the paths' associated counts, and the bottom part of the figure gives the derived outgoing edge probabilities at each vertex. This path-tree specifies, for example, that given an edge of type l was traversed from the root, the probability of reaching a correct target node is 0.9 if an edge of type n is followed, whereas the respective probability if an edge of type m is followed is estimated at a lower 0.2.

A Concurrent Graph-walk

Given a generated path tree, we apply *path-constrained* graph walks that adhere both to the topology of the graph G, and to the path tree T. Walk histories of each node x visited in the walk are compactly represented as pairs $< t, x >$, where t denotes the relevant vertex in the path tree. For example, suppose that after one walk step, the maintained node-history pairs include $< T(l), x_1 >$ and $< T(m), x_2 >$. If x_3 is reached in the next walk step

[5]The conversion to a tree is straight-forward, where identical path prefixes are merged.

[6]Another possibility is to average the downstream cumulative counts at each vertex. The MAX operation gave better results in our experiments.

```
Given: graph $G$, path-tree $T$, query distribution $V_0$, number of steps $K$
Initialize: for each $x_i \in V_0$, assign a pair $< root(T), x_i >$
Repeat for steps $k = 0$ to $K$:
For each $< t_i, x_i > \in V_k$:
Let $L$ be the set of outgoing edge labels from $x_i$, in $G$.
For each $l_m \in L$:
For each $x_j \in G$ s.t., $x_i \xrightarrow{l_m} x_j$, add $< t_j, x_j >$ to $V_{k+1}$, where $t_j \in T$, s.t. $t_i \xrightarrow{l_m} t_j$, with probability $Pr(x_i|V_k) \times Pr(l_m|t_i, T)$. (The latter probabilities should be normalized with respect to $x_i$.)
If $t_i$ is a terminal node in $T$, emit $x_i$ with probability $Pr(x_i|V_k) \times Pr(t_i|T)$.
```

Figure 3: Pseudo-code for path-constrained graph walk

from both x_1 and x_2 over paths included in the path-tree, it will be represented by multiple node pairs, e.g., $< T(l \rightarrow n), x_3 >$ and $< T(m \rightarrow l, x_3 >$. A pseudo-code for a path-constrained graph walk is given in Figure 3. It is straight-forward to discard paths in T that are associated with a lower probability than some threshold. A threshold of 0.5, for example, implies that only paths that led to a majority of positively labelled nodes in the training set are followed.

6 Related Work

Graph walks over typed graphs have been applied to derive semantic similarity for NLP problems using WordNet as a primary information source. For instance, Hughes and Ramage (2007) constructed a graph which represented various types of word relations from WordNet, and compared random-walk similarity to similarity assessments from human-subject trials. Random-walk similarity has also been used for lexical smoothing for prepositional word attachment (Toutanova et al., 2004) and query expansion (Collins-Thompson and Callan, 2005). In contrast to these works, our graph representation describes parsed text and has not been (consciously) engineered for a particular task. Instead, we include learning techniques to optimize the graph-walk based similarity measure. The learning methods described in this paper can be readily applied to other directed and labelled entity-relation graphs.[7]

The graph representation described in this paper is perhaps most related to syntax-based vector space models, which derive a notion of semantic similarity from statistics associated with a parsed corpus (Grefenstette, 1994; Lin, 1998; Padó and Lapata, 2007). In most cases, these models construct vectors to represent each word w_i, where each element in the vector for w_i corresponds to particular "context" c, and represents a count or an indication of whether w_i occurred in context c. A "context" can refer to simple co-occurrence with another word w_j, to a particular syntactic relation to another word (e.g., a relation of "direct object" to w_j), etc. Given these word vectors, inter-word similarity is evaluated using some appropriate similarity measure for the vector space, such as cosine vector similarity, or *Lin's similarity* (Lin, 1998).

Recently, Padó and Lapata (Padó and Lapata, 2007) have suggested an extended syntactic vector space model called *dependency vectors*, in which rather than simple counts, the components of a word vector of contexts consist of *weighted scores*, which combine both co-occurrence frequency and the importance of a context, based on properties of the connecting dependency paths. They considered two different weighting schemes: a *length* weighting scheme, assigning lower weight to longer connecting paths; and an *obliqueness* weighting hierarchy (Keenan and Comrie, 1977), assigning higher weight to paths that include grammatically salient relations. In an evaluation of word pair similarity based on statistics from a corpus of about 100 million words, they show improvements over several previous vector space models. Below we will compare our framework to that of Padó and Lapata. One important difference is that while Padó and Lapata make manual choices (regarding the set of paths considered and the weighting scheme), we apply learning to adjust the analogous parameters.

7 Extraction of Coordinate Terms

We evaluate the text representation schema and the proposed set of graph-based similarity measures on the task of *coordinate term* extraction. In particular, we evaluate the extraction of named entities, including *city names* and *person names* from newswire data, using word similarity measures. Coordinate terms reflect a particular type of word similarity (relatedness), and are therefore an appropriate test case for our framework. While coordinate term extraction is often addressed by a rule-based (templates) approach (Hearst, 1992), this approach was designed for very large corpora such as the Web, where the availability of many redundant documents allows use of high-precision and low-recall rules. In this paper we focus on relatively small corpora. Small limited text collections may correspond to documents residing on a personal desktop, email collections, discussion groups and other specialized sets of documents.

The task defined in the experiments is to retrieve a ranked list of city or person names given a small set of seeds. This task is implemented in the graph as a query, where we let the query distribution V_q be uniform over the given seeds (and zero elsewhere). Ideally, the resulting ranked list will be populated with many additional city, or person, names.

We compare graph walks to *dependency vectors* (DV) (Padó and Lapata, 2007),[8] as well as to a vector-based bag-of-words co-occurrence model. DV is a state-of-the-art syntactic vector-based model (see Section 6). The co-occurrence model represents a more traditional approach, where text is processed as a stream rather than syntactic structures. In applying the vector-space based methods, we compute a similarity score between *every* candidate from the corpus and each of the query terms, and then average these scores (as the query distributions are uniform) to construct a ranked list. For efficiency, in the vector-based models we limit the considered set of candidates to named entities. Similarly, the graph walk results are filtered to include named entities.[9]

Corpora. As the experimental corpora, we use the training set portion of the MUC-6 dataset (MUC, 1995) as well as articles from the Associated Press (AP) extracted from the AQUAINT corpus (Bilotti

[7]We refer the reader to the TextGraph workshop proceedings, http://textgraphs.org.

[8]We used the code from http://www.coli.uni-saarland.de/ pado/dv.html, and converted the underlying syntactic patterns to the Stanford dependency parser conventions.

[9]In general, graph walk results can be filtered by various word properties, e.g., capitalization pattern, or part-of-speech.

Corpus	*words*	*nodes*	*edges*	*unique NEs*
MUC	140K	82K	244K	3K
MUC+AP	2,440K	1,030K	3,550K	36K

Table 1: Corpus statistics

et al., 2007), all parsed using the Stanford dependency parser (de Marneffe et al., 2006).[10] The MUC corpus provides true named entity tags, while the AQUAINT corpus includes automatically generated, noisy, named entity tags. Statistics on the experimental corpora and their corresponding graph representation are detailed in Table 1. As shown, the MUC corpus contains about 140 thousand words, whereas the MUC+AP experimental corpus is substantially larger, containing about 2.5 million words.

We generated 10 queries, each comprised of 4 city names selected randomly according to the distribution of city name mentions in MUC-6. Similarly, we generated a set of 10 queries that include 4 person names selected randomly from the MUC corpus. (The MUC corpus was appended to AP, so that the same query sets are applicable in both cases.) For each task, we use 5 queries for training and tuning and the remaining queries for testing.

8 Experimental Results

Experimental setup. We evaluated cross-validation performance over the training queries in terms of mean average precision for varying walk lengths K. We found that beyond $K = 6$ improvements were small (and in fact deteriorated for $K = 9$). We therefore set $K = 6$. Weight tuning was trained using the training queries and two dozens of target nodes overall. In reranking, we set a feature count cutoff of 3, in order to avoid over-fitting. Reranking was applied to the top 200 ranked nodes output by the graph walk using the tuned edge weights. Finally, path-trees were constructed using the top 20 correct nodes and 20 incorrect nodes retrieved by the uniformly weighted graph walk. In the experiments, we apply a threshold of 0.5 to the path constrained graph walk method.

We note that for learning, true labels were used for the fully annotated MUC corpus (we hand labelled all of the named entities of type location in the corpus as to whether they were city names). However, noisy negative examples were considered for the larger automatically annotated AP corpus. (Specifically, for cities, we only considered city names included in the MUC corpus as correct answers.)

A co-occurrence vector-space model was applied using a window of two tokens to the right and to the left of the focus word. Inter-word similarity was evaluated in this model using cosine similarity, where the underlying co-occurrence counts were normalized by log-likelihood ratio (Padó and Lapata, 2007). The parameters of the DV method were set based on a cross validation evaluation (using the MUC+AP corpus). The *medium* set of dependency paths and the *oblique* edge weighting scheme were found to perform best. We experimented with cosine as well as Lin similarity measure in combination with the dependency vectors representation. Finally, given the large number of candidates in the MUC+AP corpus (Table 1), we show the results of applying the considered vector-space models to the top, high-quality, entities retrieved with reranking for this corpus.[11]

Test set results. Figure 4 gives results for the city name (top) and the person name (bottom) extraction tasks. The left part of the figure shows results using the MUC corpus, and its right part – using the MUC+AP corpus. The curves show precision as a function of rank in the ranked list, up to rank 100. (For this evaluation, we hand-labeled all the top-ranked results as to whether they are city names or person names.) Included in the figure are the curves of the graph-walk method with uniform weights (G:Uw), learned weights (G:Lw), graph-walk with reranking (Rerank) and a path-constrained graph-walk (PCW). Also given are the results of the co-occurrence model (CO), and the syntactic vector-space DV model, using the Lin similarity measure (DV:Lin). Performance of the DV model using cosine similarity was found comparable or inferior to using the Lin measure, and is omitted from the figure for clarity.

Several trends can be observed from the results. With respect to the graph walk methods, the graph walk using the learned edge weights consistently outperforms the graph walk with uniform weights. Reranking and the path-constrained graph walk,

[10]http://nlp.stanford.edu/software/lex-parser.shtml; sentences longer than 70 words omitted.

[11]We process the union of the top 200 results per each query.

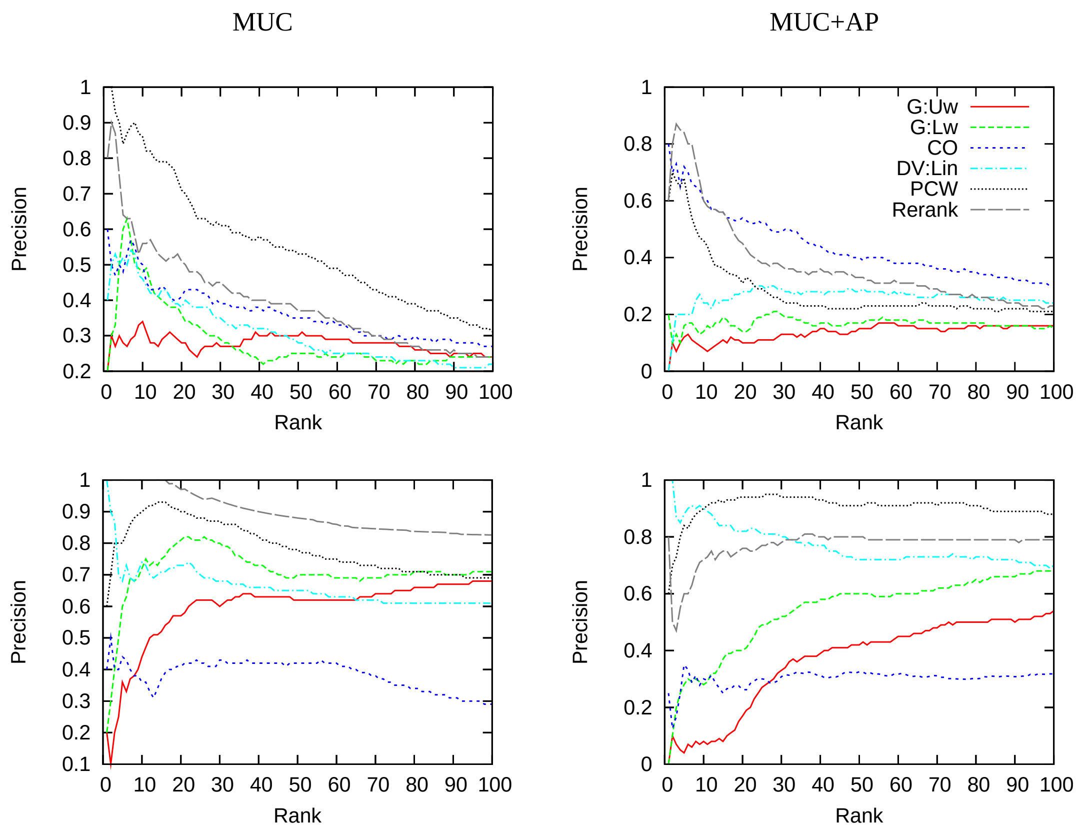


Figure 4: Test results: Precision at the top 100 ranks, for the city name extraction task (top) and person name extraction task (bottom).

however, yield superior results. Both of these learning methods utilize a richer set of features than the graph walk and weight tuning, which can consider only local information. In particular, while the graph walk paradigm assigns lower importance to longer connecting paths (as described in Section 3), reranking and the path-constrained walker allow to discard short yet irrelevant paths, and by that eliminate noise at the top ranks of the retrieved list. In general, the results show that edge sequences carry additional meaning compared with the individual edge label segments traversed.

Out of the vector-based models, the co-occurrence model is preferable for the city name extraction task, and the syntactic dependency vectors model gives substantially better performance for person name extraction. We conjecture that city name mentions are less structured in the underlying text. In addition, the syntactic weighting scheme of the DV model is probably not optimal for the case of city names. For example, a *conjunction* relation was found highly indicative for city names (see below). However, this relation is not emphasized by the DV weighting schema. As expected, the performance of the vector-based models improves for larger corpora (Terra and Clarke, 2003). These models demonstrate good performance for the larger MUC+AP corpus, but only mediocre performance for the smaller MUC corpus.

Contrasting the graph-based methods with the vector-based models, the difference in performance in favor of reranking and PCW, especially for the smaller corpus, can be attributed to two factors. The first factor is learning, which optimizes performance for the underlying data. A second factor is the incorporation of non-local information, encoding properties of the traversed paths.

Models. Following is a short description of the models learned by the different methods and tasks. Weight tuning assigned high weights to edge types such as *conj-and*, *prep-in* and *prep-from*, *nn*, *appos* and *amod* for the city extraction task. For per-

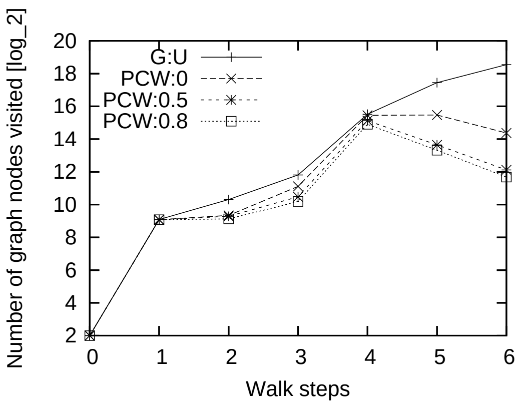


Figure 5: The graph walk exponential spread is bounded by the path constrained walk.

son extraction, prominent edge types included *subj*, *obj*, *poss* and *nn*. (The latter preferences are similar to the linguistically motivated weights of DV.) High weight features assigned by reranking for city name extraction included, for example, lexical features such as "based" and "downtown", and edge bigrams such as "prep-in-Inverse→conj-and" or "nn-Inverse→nn". Positive highly predictive paths in the constructed path tree included many symmetric paths, such as ...→conj_andInverse...→.conj_and..., ...→prep_inInverse...→.prep_in..., for the city name extraction task.

Scalability. Figure 5 shows the number of graph nodes maintained in each step of the graph walk (logarithm scale) for a typical city extraction query and the MUC+AP corpus. As shown by the solid line, the number of graph nodes visited using the weighted graph walk paradigm grows exponentially with the length of the walk. Applying a path-constrained walk with a threshold of 0 (PCW:0) reduces the maximal number of nodes expanded (as paths not observed in the training set are discarded). As shown, increasing the threshold leads to significant gains in scalability. Overall, query processing time averaged at a few minutes, using a commodity PC.

9 Conclusion and Future Directions

In this paper we make several contributions. First, we have explored a novel but natural representation for a corpus of dependency-parsed text, as a labelled directed graph. We have evaluated the task of coordinate term extraction using this representation, and shown that this task can be performed using similarity queries in a general-purpose graph-walk based query language. Further, we have successfully applied learning techniques that tune weights assigned to different dependency relations, and re-score candidates using features derived from the graph walk.

Another orthogonal contribution of this paper is a path-constrained graph walk variant, where the graph walk is guided by high level knowledge about meaningful paths, learned from training examples. This method was shown to yield improved performance for the suggested graph representation, and improved scalability compared with the local graph walk. The method is general, and can be readily applied in similar settings.

Empirical evaluation of the coordinate term extraction task shows that the graph-based framework performs better than vector-space models for the smaller corpus, and comparably otherwise. Overall, we find that the suggested model is suitable for deep (syntactic) processing of small specialized corpora. In preliminary experiments where we evaluated this framework on the task of extracting general word synonyms, using a relatively large corpus of 15 million words, we found the graph-walk performance to be better than DV using cosine similarity measures, but second to DV using Lin's similarity measure. While this set of results is incomplete, we find that it is consistent with the results reported in this paper.

The framework presented can be enhanced in several ways. For instance, WordNet edges and morphology relations can be readily encoded in the graph. We believe that this framework can be applied for the extraction of more specialized notions of word relatedness, as in relation extraction (Bunescu and Mooney, 2005). The path-constrained graph walk method proposed may be enhanced by learning edge probabilities, using a rich set of features. We are also interested in exploring a possible relation between the path-constrained walk approach and reinforcement learning.

Acknowledgments

The authors wish to thank the anonymous reviewers and Hanghang Tong for useful advice. This material is based upon work supported by Yahoo! Research.

References

Matthew W. Bilotti, Paul Ogilvie, Jamie Callan, and Eric Nyberg. 2007. Structured retrieval for question answering. In *SIGIR*.

Razvan C. Bunescu and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *HLT-EMNLP*.

William W. Cohen and Einat Minkov. 2006. A graph-search framework for associating gene identifiers with documents. *BMC Bioinformatics*, 7(440).

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–69.

Kevyn Collins-Thompson and Jamie Callan. 2005. Query expansion using random walk models. In *CIKM*.

Michael Collins. 2002. Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *ACL*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC*.

Michelangelo Diligenti, Marco Gori, and Marco Maggini. 2005. Learning web page scores by error back-propagation. In *IJCAI*.

Gregory Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers, Dordrecht.

Marti Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING*.

Thad Hughes and Daniel Ramage. 2007. Lexical semantic relatedness with random graph walks. In *EMNLP*.

Edward Keenan and Bernard Comrie. 1977. Noun phrase accessibility and universal grammar. *Linguistic Inquiry*, 8.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *COLING-ACL*.

Einat Minkov and William W. Cohen. 2007. Learning to rank typed graph walks: Local and global approaches. In *WebKDD/KDD-SNA workshop*.

Einat Minkov, William W. Cohen, and Andrew Y. Ng. 2006. Contextual search and name disambiguation in email using graphs. In *SIGIR*.

1995. Proceedings of the sixth message understanding conference (muc-6). In *Morgan Kaufmann Publishers, Inc. Columbia, Maryland.*

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2).

Philip Resnik and Mona Diab. 2000. Measuring verb similarity. In *CogSci*.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *NIPS*.

Egidio Terra and C. L. A. Clarke. 2003. Frequency estimates for statistical word similarity measures. In *NAACL*.

Kristina Toutanova, Christopher D. Manning, and Andrew Y. Ng. 2004. Learning random walk models for inducing word dependency distributions. In *ICML*.

A graph-theoretic model of lexical syntactic acquisition

Hinrich Schütze and Michael Walsh
Institute for Natural Language Processing
University of Stuttgart, Germany
{hs999,walsh}@ifnlp.org

Abstract

This paper presents a graph-theoretic model of the acquisition of lexical syntactic representations. The representations the model learns are non-categorical or graded. We propose a new evaluation methodology of syntactic acquisition in the framework of exemplar theory. When applied to the CHILDES corpus, the evaluation shows that the model's graded syntactic representations perform better than previously proposed categorical representations.

1 Introduction

In recent years, exemplar theory has had great explanatory success in phonetics. Exemplar theory posits that linguistic production and perception are not mediated via abstract categories, but that instead each production and perception of a linguistic unit is stored and retained. Linguistic inference then directly operates on these stored *exemplars*. In this paper, we propose a new approach to lexical syntactic acquisition in the framework of exemplar theory.

Our approach uses an evaluation measure that is different from previous work. Lexical syntactic acquisition is most often evaluated with respect to standard syntactic categories like verb and noun. Our first contribution in this paper is that we instead evaluate learned representations in the context of a syntactic task. This task is the determination of an aspect of grammaticality that we call *local syntactic coherence*.

Our second contribution is a *graph-theoretic model of the acquisition of lexical syntactic representations* that is more rigorous than previous heuristic proposals. The graph-theoretic model can learn both categorical and non-categorical (or graded) representations. The model is also a unified framework for syntagmatic and paradigmatic relations (as will be discussed below), and for lower-order syntactic relations (those that can be directly observed from the input) and higher-order syntactic relations (those that require some generalization from what is directly observable).

Redington et al. (1998) give an influential account of the acquisition of lexical syntactic representations in which a standard syntactic category like verb or noun is assigned to each word. Our third contribution is to show that, in the context of acquisition, *graded representations are superior to standard categorical representations* in supporting judgments of local syntactic coherence. A graded representation formalism is one that, for any two words, can represent a third word whose syntactic properties are intermediate between the two words (Manning and Schütze, 1999).

Clearly exemplar theory is not the only framework in which lexical acquisition has been explored. Gleitman (1990) for example argues for syntactic bootstrapping to infer lexical semantics, work not at odds with our own (see discussion on the role of semantics below). Our argument for the importance of distributional evidence does not call into question the large body of work in child language acquisition that demonstrates that "part of the capacity to learn languages must be 'innate' " (Gleitman and Newport, 1995). Tabula rasa learning is not possible. Our goal is not to show that language acquisition proceeds with a minimum of inductive bias. Rather, we attempt to formalize one aspect of language acquisition, the use of distributional information.

The paper is organized as follows. Section 2 motivates the exemplar-theoretic approach by reviewing its success in phonetics. Section 3 defines local syntactic coherence, which is the basis for a new evaluation methodology for the acquisition of lexical representations. Section 4 develops the graph-theoretic model. Section 5 compares graded and categorical representations for the task of inferring local syntactic coherence. Section 6 presents our evaluation. Sections 7 and 8 discuss related and future work, and present our conclusions.

2 Exemplar theory

The general idea of research into exemplars in speech production and perception is that encountered items (segments, words, sentences etc.) are stored in great detail in memory along with rich linguistic and extra-linguistic context information. These exemplars are organized into clouds of memory traces with similar traces lying close to each other while dissimilar traces are more distant. A number of such models have had great success in accounting for production and perception phenomena in phonetics. E.g., Johnson (1997) offers an exemplar model which challenges the notion that speech is perceived through a process of normalization whereby a speaker-specific representation is mapped or normalized into a speaker-neutral categorical abstraction. Johnson's model successfully treats aspects of vowel perception, sex identification, and speaker variability. Crucially, no normalization of percepts into categorical representations takes place. The correct identification of phonemes and words in his model is a function of direct comparison to richly detailed exemplars stored in memory. Other examples of exemplar-theoretic phonetic accounts include (Goldinger, 1997), (Pierrehumbert, 2001), and our own work (Schütze et al., 2007). Exemplar theory's success in phonetics motivates us to investigate its use as a model for local syntactic phenomena.

3 Local syntactic coherence

In the context sequence model for exemplar-theoretic phonetics (Wade et al., 2008), we represent speech using amplitude envelopes derived from the acoustic signal and then compute similarity as the integral over the correlation of the two acoustic signals.

For the syntactic level, we need a representation that has two key properties of the representation we use in phonetics in order to support an exemplar-theoretic account. First, the representation must be directly derivable from the perceived input. In particular, it cannot rely on the results of any disambiguation that would occur either as part of exemplar-theoretic perception or in further downstream processing. Second, it must support similarity computations. Accordingly, we first motivate the representation we use and then introduce a similarity measure on these representations.

Representation. There are two main sources[1] of directly observable information about the syntactic properties of words: semantic cues (e.g., things are often referred to with nouns) and the neighbors of a word in sentences that it is used in. In this paper, we only consider the second source of information for acquisition, lexical neighbors.[2] We further limit ourselves to the immediate left and right lexical neighbors (see discussion in Section 7).

When using lexical neighbors as the basis of representation, we have to make a basic choice as to whether we look at left and right neighbors separately or whether we only look at the "correlated" neighborhood information of left and right neighbors jointly. Our approach is based on the first alternative: we separate the processing of left and right neighbors. We do this for two reasons. First, generalization improves and model complexity decreases if left-neighbor information and right-neighbor information are looked at separately. E.g., the right neighbors of *to*, *might* and *not* are similar because all three words can be followed by base verbs like *dance*: *to dance*, *might dance*, *(might) not dance*. But their left neighbors are very different.

Second, exemplar-theoretic similarity is best defined at the smallest possible scale in order to allow optimal matching between parts of the stimulus and parts of memory. In phonetics, we use a time scale of 10s of milliseconds or even less. Conceivably, one could also use segments (e.g., consonants and vowels) as the smallest unit; however, this would presume a segmented signal. And segmentation is part of the perception task we want to explain in the first place.

Separating left and right neighbors – which amounts to looking at left and right local contexts of each word separately – is the smallest scale we can operate at when doing syntactic matching. We

[1]A comprehensive account of acquisition must also include morphology. See Christiansen et al. (2004).

[2]Psycholinguistic evidence for the importance of neighbor information for learning categories includes (Mintz, 2002).

choose this small scale for the same reasons as we choose a small scale in phonetics: to ensure maximum flexibility when matching parts of the stimulus with exemplars in memory. Using words, bigrams or larger units would reduce the flexibility in matching and require a larger amount of experience (or training data) to learn a particular generalization.

We refer to the representations of left and right contexts of a given word as *half-words*. In other words, we split a word into two entities, a left half-word that characterizes its behavior to the left and a right half-word that characterizes its behavior to the right. Thus left-context and right-context components of the representation of a given focus word are defined, where a left (right) half-word consists of a probability distribution over all words that occur to the left (right) of the focus word and the dimensionality of the vector for each word is dependent on the number of distinct neighbors (left and right). For example, having experienced *take doll* twice and *drop doll* once, then the left context distribution, or left half-word of *doll*, *doll*$_l$, is $P(\text{take}) = 2/3, P(\text{drop}) = 1/3$. By extension, the phrase *take the doll* is represented as the following six half-words: *take*$_l$, *take*$_r$, *the*$_l$, *the*$_r$, *doll*$_l$, and *doll*$_r$.

Distance measure. The basic intuition behind local syntactic coherence is that an important component of syntactic wellformedness – and a component that is of particular importance in acquisition – is whether a similar sequence has already been stored as grammatical in memory. The same way that a phonetic signal that is well-formed in a particular language has many similar exemplars in memory, a syntactic sequence should also be licensed by similar, previously perceived sequences in memory. To operationalize this notion, we need to be able to compute the similarity or distance between an input stimulus and exemplars in memory. We do this by first defining a distance measure for sequences of fixed length.

The distance Δ between two sequences of half-words $< g_1, \ldots, g_n >$ and $< h_1, \ldots, h_n >$ is defined to be the sum of the distances of their half-words:
$\Delta(<g_1, \ldots, g_n>,<h_1, \ldots, h_n>) = \sum_{i=1}^{n} \Delta(g_i, h_i)$
This definition presupposes a definition of the distance of two half-words which will be given below.

We then call a sequence of n half-words $g_1, \ldots, g_n$ *locally coherent* if there is a sequence $h_1, \ldots, h_n$ in memory with $\Delta(< g_1, \ldots, g_n >, < h_1, \ldots, h_n >) < \theta$ where θ is a parameter.

Finally, we define a sentence to be *locally n-coherent* if all of its subsequences of length n are locally coherent.

The graph-theoretic model that is introduced in the next section will be evaluated with respect to how well it captures local syntactic coherence. This enables us to evaluate the model with respect to a task as opposed to its ability to reproduce a particular linguistic representation of syntactic categories.[3] Obviously, the notion of local syntactic coherence only captures some aspects of syntax – e.g., it does not capture long-distance dependencies. However, it is a plausible component of syntactic competence and a plausible intermediate step in the acquisition of syntax.

4 Graph-theoretic model

We briefly review the structuralist notions of syntagmatic and paradigmatic relationships that have been frequently used in prior work in NLP (e.g., (Church et al., 1994)). De Saussure defined a syntagmatic relationship between two words as their contiguous occurrence in a sentence and a paradigmatic relationship as mutual substitutability (de Saussure, 1962) (although he used the term *rapport associatif* instead of *paradigmatic*). E.g., *brown* and *dog* stand in a syntagmatic relationship with each other in the phrase *brown dog*; *brown* and *black* stand in a paradigmatic relationship with each other with respect to the position between *the* and *dog* in the phrase *the X dog*. De Saussure's conceptualization of syntactic relationships captures the fact that both admissible *neighbors* and admissible *substitutes* in language are an important part of the characterization of the syntactic properties of a word.

We formalize the two relations as *distributions over words*, where we assume a vocabulary $\{w_1, \ldots, w_V\}$ and V is the number of words in the vocabulary.

We denote the *left syntagmatic distribution* of w_i

[3]Freudenthal et al. (2004) have much the same motivation in introducing an evaluation measure of syntactic acquisition based on chunking.

by $p_{i,s,l,m}$ where i is the vocabulary index of w_i, s stands for *syntagmatic*, l for *left* and m is the order of the distribution as discussed below. Intuitively, $p_{i,s,l,m}(w_j)$ is the probability that word w_j occurs to the left of w_i. Similarly, for the *left paradigmatic distribution* of w_i, $p_{i,p,l,m}(w_j)$ is the probability that w_j can be substituted for w_i without changing local syntactic coherence as far as the context to the left is concerned. Note that we distinguish between left and right paradigmatic distributions. A word w_j can be a perfect substitute for w_i as far as the context to the left is concerned, but a very unlikely substitute as far as the context to the right is concerned. E.g., in the phrase *She loves her job*, the word *him* is a good left-context substitute for *her*, but a terrible right-context substitute for *her*.

We will now show how the syntagmatic/paradigmatic (henceforth: syn/para) distributions are defined iteratively, based on the bigram distribution p_{ww}, and grounded by defining $p_{i,p,l,1}$ and $p_{i,p,r,1}$.

$p_{ww}(w_iw_j)$ is the probability that the bigram w_iw_j occurs, that is, that w_i and w_j occur next to each other (and in that order). We define the $V \times V$ *joint probability matrix* J by $J_{ij} = p_{ww}(w_iw_j)$.

Denote by N the diagonal $V \times V$ matrix that contains in N_{ii} the reciprocal of $p_w(w_i)$ where p_w is the marginal distribution of p_{ww}:

$$\sum_{j=1}^{V} p_{ww}(w_iw_j) = \sum_{j=1}^{V} p_{ww}(w_jw_i) = p_w(w_i) = \frac{1}{N_{ii}}$$

The conditional probability p_{left} of the following word and the conditional probability p_{right} of the preceding word can be computed by multiplying (the transpose of) J and N: $p_{\text{left}}(w_i|w_j) = p_{ww}(w_iw_j)/p_w(w_j) = (JN)_{ij}$; and $p_{\text{right}}(w_i|w_j) = (J^TN)_{ij}$.

The "grounding" paradigmatic distributions of order 1 are defined as follows.

$$p_{i,p,l,1}(w_j) = p_{i,p,r,1}(w_j) = \begin{cases} 0 & \text{if } w_i \neq w_j \\ 1 & \text{if } w_i = w_j \end{cases}$$

In other words, each word has only one perfect left / right substitute and that perfect substitute is itself. We define the syn/para distributions of higher order recursively:

$$p_{i,s,l,m} = JNp_{i,p,l,m} \qquad (1)$$

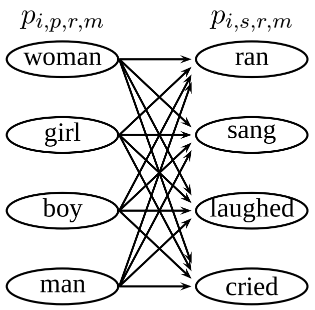


Figure 1: The distribution of typical right neighbors (the right syntagmatic distribution $p_{i,s,r,m}$) is computed from the distribution of typical "right substitutes" (the right paradigmatic distribution $p_{i,p,r,m}$).

$$p_{i,p,l,m} = J^TNp_{i,s,l,m-1} \qquad (2)$$
$$p_{i,s,r,m} = J^TNp_{i,p,r,m} \qquad (3)$$
$$p_{i,p,r,m} = JNp_{i,s,r,m-1} \qquad (4)$$

Basic matrix arithmetic shows that $p_{i,s,l,1}$ is simply $p_{\text{left}}(.|w_i)$ and $p_{i,s,r,1}$ is $p_{\text{right}}(.|w_i)$.

For higher orders, the principle underlying Eq.s 1–4 is that when moving from left to right, we use p_{right} (that is, J^TN), the conditional distribution that characterizes right neighbors; when moving from right to left, we use p_{left} (that is, JN), the conditional distribution that characterizes left neighbors. This is graphically shown in Fig. 1.

As illustrated by Fig. 1, the underlying graph for $p_{i,s,r,m}$ and $p_{i,p,r,m}$ is a weighted bipartite directed graph that connects the vocabulary on the left with the vocabulary on the right. A directed edge from w_i on the left to w_j on the right is weighted with $p_{ww}(w_iw_j)/p_w(w_i)$. A directed edge from w_j on the right to w_i on the left (not shown) is weighted with $p_{ww}(w_iw_j)/p_w(w_j)$.

Eq.s 1–4 define four Markov chains:

$$p_{i,s,l,m} = (JNJ^TN)p_{i,s,l,m-1} \qquad (5)$$
$$p_{i,p,l,m} = (J^TNJN)p_{i,p,l,m-1} \qquad (6)$$
$$p_{i,s,r,m} = (J^TNJN)p_{i,s,r,m-1} \qquad (7)$$
$$p_{i,p,r,m} = (JNJ^TN)p_{i,p,r,m-1} \qquad (8)$$

It is easy to see that p_w is a stationary distribution for Eq. 1–4. Writing $\vec{x}$ for p_w, we have:

$$(JN\vec{x})_i = \sum_{j=1}^{V} \frac{p_{ww}(w_iw_j)}{p_w(w_j)} p_w(w_j) = p_w(w_i) = x_i$$

$$(J^TN\vec{x})_i = \sum_{j=1}^{V} \frac{p_{ww}(w_jw_i)}{p_w(w_j)} p_w(w_j) = p_w(w_i) = x_i$$

Hence, p_w is a solution for Eq.s (5)–(8).

The series converge if JNJ^TN and J^TNJN are ergodic, i.e., if the chain is aperiodic and irreducible (Kemeny and Snell, 1976). Observe that for many simple probabilistic context-free grammars (PCFGs) the series in Eq. 1–4 will *not* converge. For simple PCFGs, the alternation between syntagmatic and paradigmatic distributions is periodic. E.g., if inflected verb forms only occur after nouns and nouns only before inflected verb forms, then the right syntagmatic distributions of nouns will have non-zero activation only for verbs and the right paradigmatic distributions of nouns will have non-zero activation only for nouns, thus preventing convergence.[4]

The key difference between a simple PCFG and natural language is ambiguity and noise. Because of ambiguity and noise, JNJ^TN and J^TNJN are likely to be ergodic – there is always a small non-zero probability that two words can occur next to each other. Ambiguity and noise have the same effect as teleportation for PageRank (Brin and Page, 1998) in the sense that we can jump from each word to each other word with non-zero probability.

Assuming that the Markov chains are ergodic, all four converge to p_w: $p_{i,p,r,\infty} = p_{i,p,l,\infty} = p_{i,s,r,\infty} = p_{i,s,l,\infty} = p_w$, for $1 \leq i \leq V$.

Thus, in this formalization, given enough iterations, syntagmatic and paradigmatic distributions of words eventually all become identical with the prior distribution p_w. This is surprising because linguistically and computationally syntagmatic and paradigmatic relations are fundamentally different.

However, on closer inspection, we observe that limiting the number of iterations is often beneficial when computing solutions to a problem iteratively. E.g., the expectation-maximization algorithm is often stopped early because results close to convergence are worse than results obtained after a small number of iterations. From the point of view of modeling human language acquisition, early stopping is perhaps also more realistic since humans are unlikely to perform a large number of iterations.

[4]However, non-ergodicity of JN does not imply non-ergodicity of JNJ^TN and J^TNJN, so Eq. (5)–(8) can converge even for non-ergodic JN.

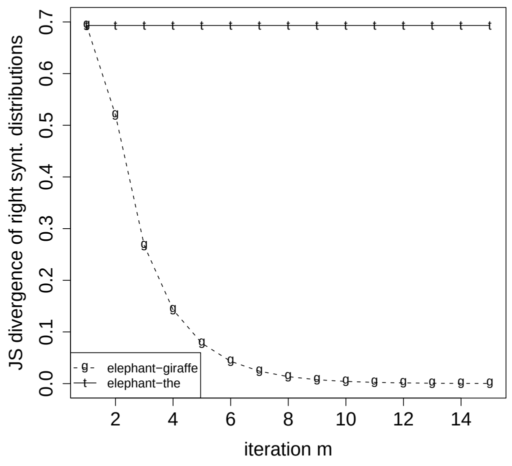


Figure 2: The distance between *elephant* and *giraffe* (measured by the Jensen-Shannon divergence) is accurately represented after a number of iterations. The words *elephant* and *the* retain their large distance.

Example 1. For the following matrix J

$$\left(\begin{array}{cccc} & w_1 & w_2 & w_3 \\ w_1 & 82/1002 & 77/1002 & 112/1002 \\ w_2 & 90/1002 & 18/1002 & 107/1002 \\ w_3 & 99/1002 & 120/1002 & 297/1002 \end{array}\right)$$

we get $p_{1,s,r,1} = (0.31, 0.28, 0.41)$ by computing the product $J^TNp_{1,p,r,1}$. E.g., $p_{1,s,r,1}(w_2) = p_{ww}(w_1w_2)/p_w(w_1) \cdot 1.0 = 77/(82+77+112) \approx 0.28$.

By iteration $m = 4$, the series $p_{i,s,r,m}$ (Eq. (7)) and $p_{i,p,r,m}$ (Eq. (8)) have converged to:
$p_{i,s,r,m} = p_{i,p,r,m} = (0.2704, 0.2145, 0.5149)$
for all three words w_i. One can easily verify that this is p_w. E.g., $p_w(w_1) = (82+90+99)/1002 = (82+77+112)/1002 \approx 0.27045$.

Example 2. We computed 15 iterations of syn/para distributions for the corpus: *The giraffe ran. An elephant fell. The man ran. An aunt fell. The man slept. The aunt slept.* Fig. 2 shows that the distance between the right syntagmatic distributions of *elephant* and *giraffe* is large for $m = 1$. The reason is that the two words have no right neighbors in common. The right neighbors of the two words are *ran* and *fell*. Although *ran* and *fell* have no left neighbors in common, their left neighbors have a right neighbor in common: the word *slept*. This indirect similarity information is exploited to deduce by iteration

15 that the two words are very similar with respect to their right syntactic context. In contrast, no such inference, even a very indirect one, is possible for the right contexts of *elephant* and *the*. Consequently, the distance between the two distributions remains high and unchanged with higher iterations.

In this case, the Markov chain is not ergodic and the syntagmatic and paradigmatic series (Eq.s (5)–(8)) do not converge to p_w.

5 Experimental evaluation

Recall from Section 3 that our evaluation task is to discriminate sentences that exhibit local coherence from those that do not; that sentences are represented as sequences of half-words; that syntactic coherence of a sentence is defined as all subsequences of a given length n exhibiting local coherence; and that a subsequence is locally coherent if its distance from a sequence in memory is less than θ.

These definitions can be applied to the graph model as follows. A left half-word is a left syntagmatic (or paradigmatic) distribution and a right half-word is a right syntagmatic (or paradigmatic) distribution. We compute the distance of two half-words either as the Jensen-Shannon (JS) divergence (Lin, 1991) or as $(1 - \cos(\alpha))$. JS divergence is more appropriate for the comparison of probability distributions. But the cosine is more efficient when a sparse vector is compared to a dense vector.[5] We therefore employ the cosine for the compute-intensive experiments in Section 6.

The baseline representation is the categorical representation proposed by Redington et al. (1998). A difficulty in replicating their experiments is that they use hierarchical agglomerative clustering (HAC), which eventually agglomerates all words in a single category. To circumvent the need for a stopping criterion, we represent each word as the temporal sequence of clusters it occurred in during agglomeration and define the distance of two words as the agglomeration step in which the two words are joined in a cluster. E.g., given the agglomeration sequences $\{1\}, \{1, 2\}, \{1, 2, 4\}, \{1, 2, 3, 4\}$ for w_1 and $\{4\}, \{4\}, \{1, 2, 4\}, \{1, 2, 3, 4\}$ for w_4, the distance between w_1 and w_4 is 3 since they are joined in step 3 when cluster $\{1, 2, 4\}$ is created.

For both graded (graph-theoretic) and categorical (cluster-based) representations, we need to set the parameter θ that is the boundary between locally coherent and locally incoherent sentences. This parameter gives rise to a precision-recall tradeoff. A small θ will impose strict requirements on which sequences in memory match, resulting in false negative decisions for local grammaticality. A large θ will incorrectly judge many locally incoherent sequences to be grammatical.

We will pick the optimal θ in both cases. For categorical representations, this amounts to selecting the HAC dendrogram with optimal performance. The experiment below evaluates whether grammatical and ungrammatical sentences are well separated by the proposed measure.[6]

Experiment on CHILDES. We used the well-known CHILDES database (MacWhinney, 2000), a corpus of conversations between young children and their playmates, siblings, and caretakers. In order to avoid mixing varieties of English (e.g., British English vs. American English), we selected the largest homogeneous subcorpus of CHILDES, the Manchester corpus. It contains roughly 350,000 sentences and 1.5 million words. This is a conservative estimate of the amount of child-directed speech a child would receive annually (Redington et al., 1998). All names in the corpus (i.e., all capitalized words) were replaced with a special word "_n_". A boundary symbol "_b_" was introduced to separate sentences. The representation of the corpus is then a concatenation of all its sentences. The vocabulary consists of $V = 8601$ words.

Construction of the evaluation set. We tested the ability of the two models to distinguish locally coherent vs. incoherent sentences by selecting 100 *unattested* sentences from the corpus, which were not used to train the model. We only selected unattested sentences that were not a substring of a sentence in the training corpus since, presumably, any substring of a sentence in the training corpus is locally coherent. A further constraint was that the

[5]This is so because, when computing the cosine, we can ignore all dimensions where one of the two vectors has a zero value.

[6]This evaluation of "separation" is not directly an evaluation of classification performance, but more similar to an evaluation of ranking using AUC or an evaluation of clustering using a measure like purity.

unattested sentence was not allowed to contain a word that did not occur in the training corpus, the rationale being that we want to address the problem of local coherence for known words only since unknown words present special challenges. Finally, we ensured that each unattested sentence contained a word that occurred in only one sentence type in the training corpus. In early experiments, we found that local grammatical inference for frequent words is easy as there is redundant evidence available that characterizes legal syntactic environments for frequent words. Since rare words are a key challenge in syntactic acquisition, we only selected sentences as unattested sentences that contained at least one rare word (where a rare word is defined as a word that occurs once in the training set).

100 ungrammatical sentences were generated by randomly selecting and concatenating words from the vocabulary. Ungrammatical sentences were matched in length to unattested sentences, so that both sets contained the same number of sentences of a given length. As with unattested sentences, ungrammatical sentences that were substrings of sentences in the training corpus were eliminated. As there are many more infrequent words than frequent words in the vocabulary, the construction ensured that, as with unattested sentences, infrequent words were overrepresented in ungrammatical sentences.

To summarize, our setup consists of 348,463 training sentences, 100 unattested grammatical sentences and 100 ungrammatical sentences.

The task of discriminating the 100 unattested from the 100 ungrammatical sentences cannot be solved perfectly as CHILDES contains ungrammatical sentences, a few of which were randomly selected as unattested sentences (e.g., *yes pleas*, which is missing the final letter). Similarly, one or two of the automatically generated ungrammatical sentences were actually grammatical.

Since the test set does not consist of a random sample of sentences, performance on the test set is not a direct indicator of the percentage of sentences that the model can correctly discriminate in a child's typical input. A large proportion of sentences in child input are simple 1-word, 2-word, and 3-word sentences that even simplistic models can evaluate with high accuracy. However, the test set is appropriate for a comparative evaluation of graded and categorical syntactic representations in language acquisition, which is one of the goals of the paper. Difficult sentences (those with rare words and greater length) are overrepresented in the test set as the discrimination of short sentences containing only frequent words can easily be done by simplistic models. Thus, a test set of "easy" sentences would not distinguish good models from bad models.

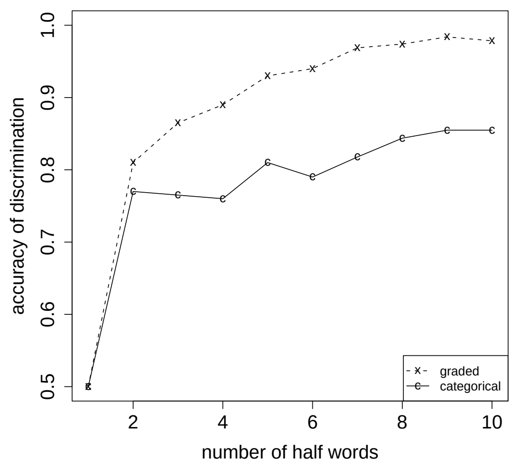


Figure 3: Accuracy of discrimination between grammatical and ungrammatical sentences for graded and categorical representations.

Discrimination experiment. In order to train the graph model, the entries of matrix J were estimated using maximum likelihood based on the training corpus. $p_{i,s,l,1}$ and $p_{i,s,r,1}$ were then computed for all 8601 words. Replicating (Redington et al., 1998), the most frequent 1000 words were clustered (using single-link HAC, Manning and Schütze (1999)). For each remaining word w, the closest neighbor w' in the 1000 most frequent words was determined and w was then assigned to the cluster of w'.

Fig. 3 shows the performance of graded and categorical representations for different subsequence sizes n. To compute the accuracy for each n, the θ with optimal discrimination performance was chosen (for both graded and categorical).

For a subsequence of size $n = 1$, the performance is 0.5 in both cases since the 200-sentence test set does not contain unknown words. So for every half-word, there is a sequence of one half-word in the training corpus with distance 0. Thus, all sentences get the same local coherence scores, both for graded and categorical representations.

This argument does not apply to $n = 2$ since we earlier defined a sentence to be locally coherent if all of its subsequences are coherent. While subsequences of 2 half-words that are part of the *same* word have local coherence score 0, this is not true of subsequences of 2 half-words that are part of *different* words, e.g., the subsequence <*black*$_r$,*dog*$_l$> in *black dog*. If *black dog* does not occur in the training set, then its local coherence score is > 0.

The main result of the experiment is that except for n=1 ($p = 1$) and n=2 ($p = 0.39$) the differences between categorical and graded representations are significant (χ^2 test, $p < 0.05$ for $3 \leq n \leq 10$). This is evidence that graded representations are more accurate when determining local syntactic coherence and grammaticality than categorical representations.

The experimental results demonstrate that, for syntagmatic distributions of order 1, graded representations discriminate locally coherent vs. incoherent sentences better than categorical representations. We attribute this to the ability of exemplar theory to incorporate rich context information into discrimination decisions. This is of particular importance for ambiguous words. Categorical representations of ambiguous words are problematic because they are either too similar or not similar enough to the two alternatives. E.g., if a word with a verb/noun ambiguity is represented as one of the alternatives, say, as a verb, then subsequences containing its noun use will no longer be similar to other subsequences with nouns. If a special conflation category noun/verb is introduced, then we are faced with the same problem: subsequences containing the noun/verb category are not similar to subsequences containing either non-ambiguous verbs or non-ambiguous nouns.

6 Higher-order distributions

The main motivation for higher-order distributions is that syntagmatic vectors of order 1 do not perform well for some infrequent words. In the elephant/giraffe example above, the distance between the two words is close to maximum for order 1 representations because each occurs only once, in entirely different contexts. As we showed in Fig. 2, higher-order representations address this problem because they exploit indirect evidence about the syntactic properties of words.

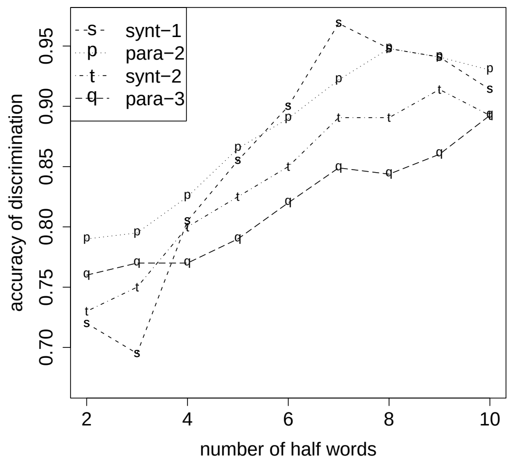


Figure 4: Accuracy of discrimination between grammatical and ungrammatical sentences of the exemplar-based method for different orders. Key: synt = syntagmatic, para = paradigmatic; s is of order 1; p and t are of order 2; q is of order 3.

To evaluate higher-order representations on CHILDES, we used the same setup as before, but computed several additional iterations. We also limited the experiments to a subset consisting of 60,000 words of the Manchester corpus. It contains only V=1666 different words, which reduces the storage requirements for the syn/para distributions (which is $2 \cdot V^2$ for each order) and the cost of the matrix multiplications. We also used $(1 - \cos(\alpha))$ instead of JS divergence as distance measure.

The results of the experiment are shown in Fig. 4. Higher-order representations are clearly superior for short subsequences, especially for $n = 2$ and $n = 3$ (and up to 5 half-words when comparing synt-1 and para-2). However, for long subsequences, there is no consistent difference between the syntagmatic distribution of order 1 (synt-1) and higher order distributions. Apparently, the generalized information available in higher orders is not helpful in local grammatical inference if long contexts are considered.

We were surprised that the best-performing distribution for short sequences is para-2 (paradigmatic distribution of order 2), not a higher order distribution. E.g., para-3 performs worse than para-2.

We would expect the performance to decrease with higher order eventually since the distributions converge towards p_w. The fact that this happens so early in this experiment merits further investigation.

7 Related work

Data-oriented parsing (Bod et al., 2003) shares basic assumptions about linguistic inference with exemplar-based theory, but it does not model or use the similarity between input and stored exemplars. Previous work on exemplar theory in syntax (Abbot-Smith and Tomasello, 2006; Bybee, 2006; Hay and Bresnan, 2006) has not been computational or formal. Previous work on non-categorical representations of words has viewed these representations as an intermediate step for arriving at categorical parts of speech (Redington et al., 1998; Schütze, 1995; Clark, 2003). Consequently, all of these papers evaluate their results by comparing induced categories to gold-standard parts of speech.

Redington et al. (1998) did not find a difference in categorization accuracy between simple syntagmatic representation and those using non-adjacent words.

The BEAGLE model (Jones and Mewhort, 2007), and related work (Sahlgren et al., 2008), merges co-occurrence information and word order information into a single composite vector through a process of vector convolution. Our model differs in that it explicitly captures the recursive relationship between the orders in a unified framework.

Previous graph-theoretic work (Biemann, 2006) uses order 1 representations. Several papers have looked at higher-order representations, but have not examined the equivalence of syn/para distributions when formalized as Markov chains (Schütze and Pedersen, 1993; Lund and Burgess, 1996; Edmonds, 1997; Rapp, 2002; Biemann et al., 2004; Lemaire and Denhière, 2006). Toutanova et al. (2004) found that their graph model of predicate argument structure deteriorated after a small number of iterations of the random walk, similar to our findings.

8 Conclusions and Future Work

In this paper, we have presented a graph-theoretic model of the acquisition of lexical syntactic representations and a new exemplar-based evaluation of lexical syntactic acquisition. When applied to the CHILDES corpus, the evaluation shows that the graded syntactic representations learned by the model perform significantly better than previously proposed categorical representations. An initial evaluation of high-order representations showed little improvement over low-order representations.

In future work, we intend to investigate the influence of noise and ambiguity on the quality of the representations in order to characterize when higher order representations improve generalization and exemplar-theoretic inference. We also want to address that the model as it currently stands is trained under the false assumption that the training input is grammatical. Ungrammatical test input which matches a learned ungrammatical sequence will be deemed grammatical. Future work will examine how to best treat this challenge, e.g., by using an estimation of density instead of the simplistic "1 nearest neighbor" distance used here.

The most important future work concerns class-based language models. The cognitive-linguistic tradition we have mainly addressed in this paper has focused on the task of learning traditional parts of speech and has usually not discussed the relevance of language models to acquisition. If, as we have argued, instead of learning traditional parts of speech the focus should be on performance in particular language processing tasks (like grammaticality judgments), then language models are the natural competing account that we must compare our work to. Of particular relevance are class-based language models (e.g., (Saul and Pereira, 1997; Brown et al., 1992)). In ongoing work, we are attempting to show that the exemplar-theoretic model performs better on grammaticality judgments than class-based language models.

Acknowledgements. This research was funded by the German Research Council (DFG, Grant SFB 732). We thank K. Rothenhäusler, H. Schmid and the reviewers for their valuable comments.

References

Abbot-Smith, Kirsten and Michael Tomasello. 2006. Exemplar-learning and schematization in a usage-based account of syntactic acquisition. *The Linguistic Review*, 23:275–290.

Biemann, Chris, Stefan Bordag, and Uwe Quasthoff. 2004. Automatic acquisition of paradigmatic relations using iterated co-occurrences. In *LREC*.

Biemann, Chris. 2006. Unsupervised part-of-speech tagging employing efficient graph clustering. In *ACL*.

Bod, Rens, Remko Scha, and Khalil Sima′an. 2003. *Data-Oriented Parsing*. CSLI Publications.

Brin, Sergey and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. In *WWW*, pages 107–117.

Brown, Peter F., Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479.

Bybee, Joan L. 2006. From usage to grammar: The mind's response to repetition. *Language*, 82:711–733.

Christiansen, Morten, Luca Onnis, Padraic Monaghan, and Nick Chater. 2004. Happy endings in language acquisition. In *AMLaP*.

Church, Kenneth, Patrick Hanks, Donald Hindle, William Gale, and Rosamund Moon. 1994. Lexical substitutability. In Atkins, B.T.S. and A. Zampolli, editors, *Computational Approaches to the Lexicon*. OUP.

Clark, Alexander. 2003. Combining distributional and morphological information for part of speech induction. In *EACL*, pages 59–66.

de Saussure, Ferdinand. 1962. *Cours de linguistique générale*. Payot, Paris. Originally published in 1916.

Edmonds, Philip. 1997. Choosing the word most typical in context using a lexical co-occurrence network. In *ACL*, pages 507–509.

Freudenthal, Daniel, Julian Pine, and Fernand Gobet. 2004. Resolving ambiguities in the extraction of syntactic categories through chunking. In *ICCM*.

Gleitman, Lila and Elissa Newport. 1995. The invention of language by children: Environmental and biological influences on the acquisition of language. In Gleitman, Lila and Mark Liberman, editors, *Language: An invitation to cognitive science*. MIT Press, 2nd edition.

Gleitman, Lila. 1990. The structural sources of verb meanings. *Language Acquisition*, 1:3–55.

Goldinger, Stephen D. 1997. Words and voices—perception and production in an episodic lexicon. In (Johnson and Mullennix, 1997).

Hay, Jennifer and Joan Bresnan. 2006. Spoken syntax: The phonetics of giving a hand in New Zealand English. *The Linguistic Review*, 23.

Johnson, Keith and John W. Mullennix, editors. 1997. *Talker Variability in Speech Processing*. Academic Press.

Johnson, Keith. 1997. Speech perception without speaker normalization. In (Johnson and Mullennix, 1997).

Jones, Michael N. and Douglas J.K. Mewhort. 2007. Representing word meaning and order information in a composite holographic lexicon. *Psychological Review*, 114:1–37.

Kemeny, John G. and J. Laurie Snell. 1976. *Finite Markov Chains*. Springer, New York.

Lemaire, Benoit and Guy Denhière. 2006. Effects of high-order co-occurrences on word semantic similarity. *Behaviour, Brain & Cognition*, 18(1).

Lin, Jianhua. 1991. Divergence measures based on the Shannon entropy. *IEEE Trans. Inf. Theory*, 37(1):145–151.

Lund, Kevin and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instrumentation, and Computers*, 28:203–208.

MacWhinney, Brian. 2000. *The CHILDES project: Tools for analyzing talk*. Lawrence Erlbaum.

Manning, Christopher D. and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Boston, MA.

Mintz, Toben H. 2002. Category induction from distributional cues in an artificial language. *Memory & Cognition*, 30:678–686.

Pierrehumbert, Janet. 2001. Exemplar dynamics: Word frequency, lenition and contrast. In Bybee, Joan and Paul Hopper, editors, *Frequency and the Emergence of Linguistic Structure*, pages 137–157. Benjamins.

Rapp, Reinhard. 2002. The computation of word associations: comparing syntagmatic and paradigmatic approaches. In *Coling*.

Redington, Martin, Nick Chater, and Steven Finch. 1998. Distributional information: A powerful cue for acquiring syntactic categories. *Cognitive Science*, 22(4):425–469.

Sahlgren, Magnus, Anders Holst, and Jussi Karlgren. 2008. Permutations as a means to encode order in word space. In *CogSci*.

Saul, Lawrence and Fernando Pereira. 1997. Aggregate and mixed-order markov models for statistical language processing. In *EMNLP*, pages 81–89.

Schütze, Hinrich and Jan Pedersen. 1993. A vector model for syntagmatic and paradigmatic relatedness. In *UW Centre for the New OED and Text Research*.

Schütze, Hinrich, Michael Walsh, Travis Wade, and Bernd Möbius. 2007. Towards a unified exemplar-theoretic model of phonetic and syntactic phenomena. In *CogSci, Poster Session*.

Schütze, Hinrich. 1995. Distributional part-of-speech tagging. In *EACL*, pages 141–148.

Toutanova, Kristina, Christopher D. Manning, and Andrew Y. Ng. 2004. Learning random walk models for inducing word dependency distributions. In *ICML*.

Wade, Travis, Grzegorz Dogil, Hinrich Schütze, Michael Walsh, and Bernd Möbius. 2008. Syllable frequency effects in a context-sensitive segment production model. Submitted.

Question Classification using Head Words and their Hypernyms

Zhiheng Huang
EECS Department
University of California
at Berkeley
CA 94720-1776, USA
zhiheng@cs.berkeley.edu

Marcus Thint
Intelligent Systems Research Center
British Telecom Group
Chief Technology Office
marcus.2.thint@bt.com

Zengchang Qin
EECS Department
University of California
at Berkeley
CA 94720-1776, USA
zqin@cs.berkeley.edu

Abstract

Question classification plays an important role in question answering. Features are the key to obtain an accurate question classifier. In contrast to Li and Roth (2002)'s approach which makes use of very rich feature space, we propose a compact yet effective feature set. In particular, we propose head word feature and present two approaches to augment semantic features of such head words using WordNet. In addition, Lesk's word sense disambiguation (WSD) algorithm is adapted and the depth of hypernym feature is optimized. With further augment of other standard features such as unigrams, our linear SVM and Maximum Entropy (ME) models reach the accuracy of 89.2% and 89.0% respectively over a standard benchmark dataset, which outperform the best previously reported accuracy of 86.2%.

1 Introduction

An important step in question answering (QA) and other dialog systems is to classify the question to the anticipated type of the answer. For example, the question of *Who discovered x-rays* should be classified into the type of human (individual). This information would narrow down the search space to identify the correct answer string. In addition, this information can suggest different strategies to search and verify a candidate answer. For instance, the classification of question *What is autism* to a definition type question would trigger the search strategy specific for definition type (e.g., using predefined templates like: *Autism is ...* or *Autism is defined as...*). In fact, the combination of QA and the named entity recognition is a key approach in modern question answering systems (Voorhees and Dang, 2005).

The question classification is by no means trivial: Simply using question wh-words can not achieve satisfactory results. The difficulty lies in classifying the *what* and *which* type questions. Considering the example *What is the capital of Yugoslavia*, it is of location (city) type, while *What is the pH scale* is of definition type. Considering also examples (Li and Roth, 2006) *What tourist attractions are there in Reims*, *What are the names of the tourist attractions in Reims*, *What do most tourists visit in Reims*, *What attracts tourists to Reims*, and *What is worth seeing in Reims*, all these reformulations are of the same answer type of location. Different wording and syntactic structures make it difficult for classification.

Many QA systems used manually constructed sets of rules to map a question to a type, which is not efficient in maintain and upgrading. With the increasing popularity of statistical approaches, machine learning plays a more and more important role in this task. A salient advantage of machine learning approach is that one can focus on designing insightful features, and rely on learning process to efficiently and effectively cope with the features. In addition, a learned classifier is more flexible to reconstruct than a manually constructed system because it can be trained on a new taxonomy in a very short time. Earlier question classification work includes Pinto et al. (2002) and Radev et at. (2002), in which language model and Rappier rule learning were employed respectively. More recently, Li and Roth (2002) have developed a machine learning approach which uses the SNoW learning architecture (Khardon et al.,

Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, pages 927–936,
Honolulu, October 2008. ©2008 Association for Computational Linguistics

1999). They have compiled the UIUC question classification dataset [1] which consists of 5500 training and 500 test questions. The questions in this dataset are collected from four sources: 4,500 English questions published by USC (Hovy et al., 2001), about 500 manually constructed questions for a few rare classes, 894 TREC 8 and TREC 9 questions, and also 500 questions from TREC 10 which serve as the test dataset. All questions in the dataset have been manually labeled by them according to the coarse and fine grained categories as shown in Table 3, with coarse classes (in bold) followed by their fine class refinements. In addition, the table shows the distribution of the 500 test questions over such categories. Li and Roth (2002) have made use of lexical words, part of speech tags, chunks (non-overlapping phrases), head chunks (the first noun chunk in a question) and named entities. They achieved 78.8% accuracy for 50 fine grained classes. With a hand-built dictionary of semantically related words, their system is able to reach 84.2%.

The UIUC dataset has laid a platform for the follow-up research. Hacioglu and Ward (2003) used linear support vector machines with question word bigrams and error-correcting output to obtain accuracy of 80.2% to 82.0%. Zhang and Lee (2003) used linear SVMs with all possible question word grams, and obtained accuracy of 79.2%. Later Li and Roth (2006) used more semantic information sources including named entities, WordNet senses, class-specific related words, and distributional similarity based categories in question classification task. With all these semantic features plus the syntactic ones, their model was trained on 21'500 questions and was able to achieve the best accuracy of 89.3% on a test set of 1000 questions (taken from TREC 10 and TREC 11) for 50 fine classes. Most recently, Krishnan et al. (2005) used a short (typically one to three words) subsequence of question tokens as features for question classification. Their model can reach the accuracy of 86.2% using UIUC dataset over fine grained question categories, which is the highest reported accuracy on UIUC dataset.

In contrast to Li and Roth (2006)'s approach which makes use of a very rich feature set, we propose to use a compact yet effective feature set. In particular, we propose head word feature and present two approaches to augment semantic features of such head words using WordNet. In addition, Lesk's word sense disambiguation (WSD) algorithm is adapted and the depth of hypernym feature is optimized. With further augment of other standard features such as unigrams, we can obtain accuracy of 89.2% using linear SVMs, or 89.0% using ME for 50 fine classes.

[1]available at http://12r.cs.uiuc.edu/∼cogcomp/Data/QA/QC

2 Classifiers

In this section, we briefly present two classifiers, support vector machines and maximum entropy model, which will be employed in our experiments. These two classifiers perform roughly identical in the question classification task.

2.1 Support Vector Machines

Support vector machine (Vapnik, 1995) is a useful technique for data classification. Given a training set of instance-labeled pairs $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, l$ where $\mathbf{x}_i \in R^n$ and $\mathbf{y} \in \{1, -1\}^l$, the support vector machines (SVM) require the solution of the following optimization problem: $\min_{\mathbf{w},b,\xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i$ subject to $y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$. Here training vectors $\mathbf{x}_i$ are mapped into a higher (maybe infinite) dimensional space by the function ϕ. Then SVM finds a linear separating hyperplane with the maximal margin in this higher dimensional space. $C > 0$ is the penalty parameter of the error term. Furthermore, $K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_i)$ is called the kernel function. There are four basic kernels: linear, polynomial, radial basis function, and sigmoid. In the question classification context, $\mathbf{x}_i$ is represented by a set of binary features, for instance, the presence or absence of particular words. $y_i \in \{1, -1\}$ indicates wether a question is of a particular type or not. Due to the large number of features in question classification, one may not need to map data to a higher dimensional space. It has been commonly accepted that the linear kernel of $K(\mathbf{x}_i, \mathbf{x}_j) = {\mathbf{x}_i}^T\mathbf{x}_i$ is good enough for question classification. In this paper, we adopt the LIBSVM (Chang and Lin, 2001) implementation in our experiments.

2.2 Maximum Entropy Models

Maximum entropy (ME) models (Berger et al., 1996; Manning and Klein, 2003), also known as log-linear and exponential learning models, provide a general purpose machine learning technique for classification and prediction which has been successfully applied to natural language processing including part of speech tagging, named entity recognition etc. Maximum entropy models can integrate features from many heterogeneous information sources for classification. Each feature corresponds to a constraint on the model. In the context of question classification, a sample feature could be the presence of a particular word associated with a particular question type. The maximum entropy model is the model with maximum entropy of all models that satisfy the constraints. In this paper, we adopt Stanford Maximum Entropy (Manning and Klein, 2003) implementation in our experiments.

3 Features

Each question is represented as a bag of features and is feeded into classifiers in training stage. We present five binary feature sets, namely question wh-word, head word, WordNet semantic features for head word, word grams, and word shape feature. The five feature sets will be separately used by the classifiers to determine their individual contribution. In addition, these features are used in an incremental fashion in our experiments.

3.1 Question wh-word

The wh-word feature is the question wh-word in given questions. For example, the wh-word of question *What is the population of China* is *what*. We have adopted eight question wh-words, namely *what*, *which*, *when*, *where*, *who*, *how*, *why*, and *rest*, with *rest* being the type does not belong to any of the previous type. For example, the question *Name a food high in zinc* is a *rest* type question.

3.2 Head Word

Li and Roth (2002;2006) used head chunks as features. The first noun chunk and the first verb chunk after the question word in a sentence are defined as head chunks in their approach. Krishnan et al. (2005) used one contiguous span of tokens which is denoted as the *informer span* as features. In both approaches, noisy information could be introduced. For example, considering the question of *What is a group of turkeys called*, both the head chunk and informer span of this question is *group of turkeys*. The word of *turkeys* in the chunk (or span) contributes to the classification of type ENTY:animal if the hypernyms of WordNet are employed (as described in next section). However, the extra word *group* would introduce ambiguity to misclassify such question into HUMAN:group, as all words appearing in chunk are treated equally. To tackle this problem, we propose the feature of *head word*, which is one single word specifying the object that the question seeks. In the previous example *What is a group of turkeys called*, the head word is exactly *turkeys*. In doing so, no misleading word *group* is augmented. Another example is *George Bush purchased a small interest in which baseball team*. The head chunk, informer span and head word are *baseball team*, *baseball team* and *team* respectively. The extra word *baseball* in the head chunk and informer span may lead the question misclassified as ENTY:sport rather than HUM:group. In most cases, the head chunk or informer span include head words. The head chunk feature or informer span feature would be beneficiary so long as the useful information plays a stronger role than the misleading one. Nevertheless, this is not as effective as the introduction of one head word.

To obtain the head word feature, a syntactic parser is required. A syntactic parser is a model that outputs the grammatical structure of given sentences. There are accurate parsers available such as Chaniak parser (Charniak and Johnson, 2005), Stanford parser (Klein and Manning, 2003) and Berkeley parser (Petrov and Klein, 2007), among which we use the Berkeley parser [2] to help identify the head word. Figure 1 shows two example parse trees for questions *What year did the Titanic sink* and *What is the sales tax in Minnesota* respectively.

Collins rules (Collins, 1999) can be applied to parse trees to extract the syntactic head words. For example, the WHNP phrase (Wh-noun phrase) in the top of Figure 1 takes its WP child as its head word, thus assigning the word *what* (in the bracket) which is associated with WP tag to the syntactic head word of WHNP phrase. Such head word assignment is carried out from the bottom up and the word *did* is extracted as the head word of the whole question. Similarly, the word *is* is extracted as the

[2] available at http://nlp.cs.berkeley.edu/Main.html#parsing

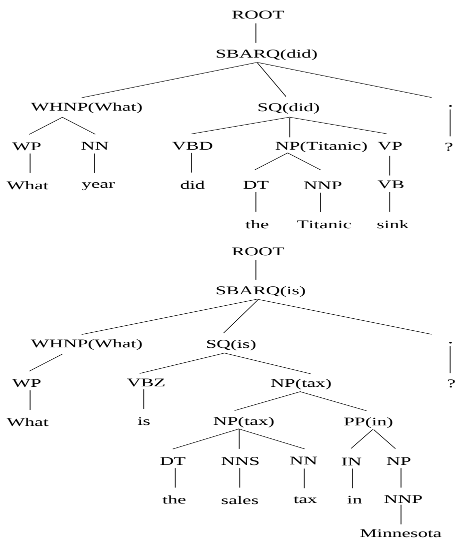


Figure 1: Two example parse trees and their head words assignment

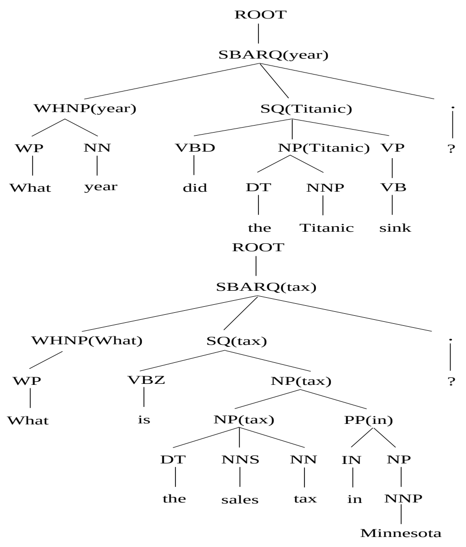


Figure 2: Two example parse trees and their revised head words assignment

syntactic head word in the bottom of Figure 1.

Collins head words finder rules have been modified to extract semantic head word (Klein and Manning, 2003). To better cover the question sentences, we further re-define the semantic head finder rules to fit our needs. In particular, the rules to find the semantic head word of phrases SBARQ (Clause introduced by subordinating conjunction), SQ (subconstituent of SBARQ excluding wh-word or wh-phrase), VP (verb phrase) and SINV (declarative sentence with subject-aux inversion) are redefined, with the head preference of noun or noun phrase rather than verb or verb phrase. The new head word assignments for the previous two examples are shown in Figure 2.

If the head word is any of *name*, *type* or *kind* etc, post fix is required to identify the real head word if necessary. In particular, we compile a tree pattern as shown in the left of Figure 3. If this pattern is matched against a given parse question parse tree, the head word is re-assigned to the head word of NP node in the tree pattern. For example, the initial head word extracted from parse tree of question *What is the proper name for a female walrus* is *name*. As such parse tree (as shown partially in the right of Figure 3) matches the compiled tree pattern, the post operation shall fix it to *walrus*, which is the head word of the NP in the tree pattern. This post fix helps classify the question to ENTY:animal.

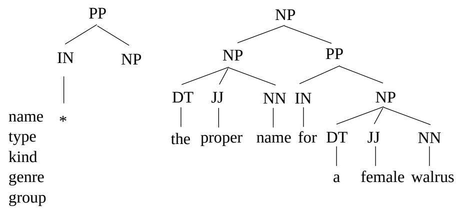


Figure 3: Post fix for the head word assignment

In addition to the question head word as described above, we introduce a few regular expression patterns to help question head word identification. Note that these patterns depend on the question type taxonomy as shown in Table 3. For example, considering the questions of *What is an atom* and *What are invertebrates*, the head word of *atom* and *invertebrates* do not help classify such questions to DESC:def. To resolve this, we create a binary feature using a string regular expression which begins with *what is/are* and follows by an optional *a*, *an*, or *the* and then follows by one or two words. If a question matches this regular expression, a binary feature (a placeholder word is used in implementation, for instance DESC:def_1 in this case) would be inserted to the feature set of the question. This feature, if it is beneficial, would be picked up by the classifiers (SVMs or MEs) in training. We list all regular expression patterns which are used in our experiments as following:

DESC:def pattern 1 The question begins with *what is/are* and follows by an optional *a*, *an*, or *the* and then follows by one or two words.

DESC:def pattern 2 The question begins with *what do/does* and ends with *mean*.

ENTY:substance pattern The question begins with *what is/are* and ends with *composed of/made of/made out of*.

DESC:desc pattern The question begins with *what does* and ends with *do*.

ENTY:term The question begins with *what do you call*.

DESC:reason pattern 1 The question begins with *what causes/cause*.

DESC:reason pattern 2 The question begins with *What is/are* and ends with *used for*.

ABBR:exp pattern The question begins with *What does/do* and ends with *stand for*.

HUM:desc pattern The question begins with *Who is/was* and follows by a word starting with a capital letter.

It is worth noting that all these patterns serve as feature generators for given questions: the feature becomes active if the pattern matches the questions. The algorithm to extract question head word is shown in Algorithm 1. There is no head word returned for *when*, *where* or *why* type questions, as these hw-words are informative enough; the inclusion of other words would introduce noisy information. If the question is of type *how*, the word following *how* is returned as head word. The patterns are then attempted to match the question if it is of type *what* or *who*. If there is a match, the placehold word for such pattern (e.g., *HUM:desc* for HUM:desc pattern) is returned as head word. If none of the above condition is met, the candidate head word is extracted from the question parse tree using the redefined head finder rules. Such extracted head word is returned only if it has noun or noun phrase tag; otherwise the first word which has noun or noun phrase tag is returned. The last step is a back up plan in case none of the previous procedure happens.

Algorithm 1 Question head word extraction

```
Require: Question $q$
Ensure: Question head word
 1: if $q$.type == $when|where|why$ then
 2:     return null
 3: end if
 4: if $q$.type == how then
 5:     return the word following word "how"
 6: end if
 7: if $q$.type == what then
 8:     for any aforementioned regular expression $r$ (except HUM:desc pattern) do
 9:         if($q$ matches $r$)
10:         return $r$.placehold-word
11:     end for
12: end if
13: if $q$.type == who && $q$ matches HUM:desc pattern then
14:     return "HUM:desc"
15: end if
16: String $candidate$ = head word extracted from question parse tree
17: if $candidate$.tag starts with NN then
18:     return $candidate$
19: end if
20: return the first word whose tag starts with NN
```

3.3 WordNet Semantic Feature

WordNet (Fellbaum, 1998) is a large English lexicon in which meaningfully related words are connected via cognitive synonyms (synsets). The WordNet is a useful tool for word semantics analysis and has been widely used in question classification (Krishnan et al., 2005; Schlaefer et al., 2007). A natural way to use WordNet is via hypernyms: Y is a hypernym of X if every X is a (kind of) Y. For example, the question of *What breed of hunting dog did the Beverly Hillbillies own* requires the knowledge of *animal* being the hypernym of *dog*. In this paper, we propose two approaches to augment WordNet semantic features, with the first augmenting the hypernyms of head words as extracted in previous section directly, and the second making use of a WordNet similarity package (Seco et al., 2004), which implicitly employs the structure of hypernyms.

3.3.1 Direct Use of Hypernyms

In WordNet, senses are organized into hierarchies with hypernym relationships, which provides a natural way to augment hypernyms features from the original head word. For example, the hierarchies for a noun sense of domestic dog is described as: *dog* → *domestic animal* → *animal*, while another noun sense (a dull unattractive unpleasant girl or woman) is organized as *dog* → *unpleasant woman* → *unpleasant person*. In addition, a verb sense of dog is organized as *dog* → *pursue* → *travel*. In our first approach, we attempt to directly introduce hypernyms for the extracted head words. The augment of hypernyms for given head word can introduce useful information, but can also bring noise if the head word or the sense of head word are not correctly identified. To resolve this, three questions shall be addressed: 1) which part of speech senses should be augmented? 2) which sense of the given word is needed to be augmented? and 3) how many depth are required to tradeoff the generality (thus more informative) and the specificity (thus less noisy). The first question can be answered by mapping the Penn Treebank part of speech tag of the given head word to its WordNet part of speech tag, which is one of POS.NOUN, and POS.ADJECTIVE, POS.ADVERB and POS.VERB. The second question is actually a word sense disambiguation (WSD) problem. The Lesk algorithm (Lesk, 1986) is a classical algorithm for WSD. It is based on the assumption that words in a given context will tend to share a common topic. A basic implementation of the The Lesk algorithm is described as following:

1. Choosing pairs of ambiguous words within a context
2. Checks their definitions in a dictionary
3. Choose the senses as to maximize the number of common terms in the definitions of the chosen words

In our head word sense disambiguation, the context words are words (except the head word itself) in the question, and the dictionary is the gloss of a sense for a given word. Algorithm 2 shows the adapted Lesk algorithm which is employed in our system. Basically, for each sense of given head word, this

Algorithm 2 Head word sense disambiguation

```
Require: Question $q$ and its head word $h$
Ensure: Disambiguated sense for $h$
 1: int $count$ = 0
 2: int $maxCount$ = -1
 3: sense $optimum$ = null
 4: for each sense $s$ for $h$ do
 5:     $count$ = 0
 6:     for each context word $w$ in $q$ do
 7:         int $subMax$ = maximum number of common words in $s$
            definition (gloss) and definition of any sense of $w$
 8:         count = count + sumMax
 9:     end for
10:     if $count > maxCount$ then
11:         $maxCount = count$
12:         $optimum = s$
13:     end if
14: end for
15: return $optimum$
```

algorithm computes the maximum number of common words between gloss of this sense and gloss of any sense of the context words. Among all head word senses, the sense which results in the maximum common words is chosen as the optimal sense to augment hypernyms later. Finally the third question is answered via trail and error based on evaluating randomly generated 10% data from the training dataset. Generally speaking, if the identification of the head word is not accurate, it would brings significant noisy information. Our experiments show that the use of depth six produces the best results over the validation dataset. This indirectly proves that our head word feature is very accurate: the hypernyms introduction within six depths would otherwise pollute the feature space.

3.3.2 Indirect Use of Hypernyms

In this approach, we make use of the WordNet Similarity package (Seco et al., 2004), which implicitly employs WordNet hypernyms. In particular, for a given pair of words, the WordNet similarity package models the length of path traveling from one word to the other over the WordNet network. It then computes the semantic similarity based on the path. For example, the similarity between *car* and *automobile* is 1.0, while the similarity between *film* and *audience* is 0.38. For each question, we use the WordNet similarity package to compute the similarity between the head word of such question and each description word in a question categorization. The description words for a question category are a few words (usually one to three) which explain the semantic meaning of such a question category [3]. For example, the descriptions words for category ENTY:dismed are *diseases* and *medicine*. The question category which has the highest similarity to the head word is marked as a feature. This is equal to a mini question classifier. For example, as the head word *walrus* of question *What is the proper name for a female walrus* has the highest similarity measure to *animals*, which is a description word of category ENTY:animal, thus the ENTY:animal is inserted into the feature set of the given question.

3.4 N-Grams

An N-gram is a sub-sequence of N words from a given question. Unigram forms the bag of words feature, and bigram forms the pairs of words feature, and so forth. We have considered unigram, bigram, and trigram features in our experiments. The reason to use such features is to provide word sense

[3]available at http://12r.cs.uiuc.edu/∼cogcomp/Data/QA/QC/definition.html

disambiguation for questions such as *How long did Rip Van Winkle sleep*, as *How long* (captured by wh-word and head word features) could refer to either NUM:dist or NUM:period. The word feature of *sleep* help determine the NUM:period classification.

3.5 Word Shape

Word shape in a given question may be useful for question classification. For instance, the question *Who is Duke Ellington* has a mixed shape (begins a with capital letter and follows by lower case letters) for *Duke*, which roughly serves as a named entity recognizer. We use five word shape features, namely all upper case, all lower case, mixed case, all digits, and other. The experiments show that this feature slightly boosts the accuracy.

4 Experimental Results

We designed two experiments to test the accuracy of our classifiers. The first experiment evaluates the individual contribution of different feature types to question classification accuracy. In particular, the SVM and ME are trained from the UIUC 5500 training data using the following feature sets: 1) wh-word + head word, 2) wh-word + head word + direct hypernym, 3) wh-wod + head word + indirect hypernym, 4) unigram, 5) bigram, 6) trigram, and 7) word shape. We set up the tests of 1), 2) and 3) due to the fact that wh-word and head word can be treated as a unit, and hypernym depends on head word. In the second experiment, feature sets are incrementally feeded to the SVM and ME. The parameters for both SVM and ME classifiers (e.g., the C in the SVM) are all with the default values. In order to facilitate the comparison with previously reported results, the question classification performance is measured by accuracy, i.e., the proportion of the correctly classified questions among all test questions.

4.1 Individual Feature Contribution

Table 1 shows the question classification accuracy of SVM and ME using individual feature sets for 6 coarse and 50 fine classes. Among all feature sets, wh-word + head word proves to be very informative for question classification. Our first WordNet semantic feature augment, the inclusion of direct hypernym, can further boost the accuracy in the fine classes for both SVM and ME, up to four per-

Table 1: Question classification accuracy of SVM and ME using individual feature sets for 6 and 50 classes over UIUC dataset

		6 class		50 class	
		SVM	ME	SVM	ME
wh-word + head word		92.0	92.2	81.4	82.0
wh-word +	depth=1	92.0	91.8	84.6	84.8
head word +	depth = 3	92.0	92.2	85.4	85.4
direct hypernym	depth = 6	92.6	91.8	85.4	85.6
wh-word + head		91.8	92.0	83.2	83.6
+ indirect hypernym					
unigram		88.0	86.6	80.4	78.8
bigram		85.6	86.4	73.8	75.2
trigram		68.0	57.4	39.0	44.2
word shape		18.8	18.8	10.4	10.4

cent. This phenomena conforms to Krishnan et al. (2005) that WordNet hypernym benefits mainly on the 50 fine classes classification. Li and Roth (2006) made use of semantic features including named entities, WordNet senses, class-specific related words, and distributional similarity based categories. Their system managed to improve around 4 percent with the help of those semantic features. They reported that WordNet didn't contribute much to the system, while our results show that the WordNet significantly boosts the accuracy. The reason may be that their system expanded the hypernyms for each word in the question, while ours only expanded the head word. In doing so, the augmentation does not introduce much noisy information. Notice that the inclusion of various depth of hypernyms results in different accuracy. The depth of six brings the highest accuracy of 85.4% and 85.6% for SVM and ME under 50 classes, which is very competitive to the previously reported best accuracy of 86.2% (Krishnan et al., 2005).

Our second proposed WordNet semantic feature, the indirect use of hypernym, does not perform as good as the first approach; it only contributes the accuracy gain of 1.8 and 1.6 in the fine classes for SVM and ME respectively. The reason may be two fold: 1) the description words (usually one to three words) of question categories are not representative enough, and 2) the indirect use of hypernyms via the WordNet similarity package is not as efficient as direct use of hypernyms.

Among the surface words features, unigram feature perform the best with accuracy of 80.4% for SVM under 50 classes, and 88.0% for SVM under 6 classes. It is not surprising that the word shape feature only achieves small gain in question classification, as the use of five shape type does not provide enough information for question classification. However, this feature is treated as an auxiliary one to boost a good classifier, as we will see in the second experiment.

4.2 Incremental Feature Contribution

Based on the individual feature contribution, we then trained the SVMs and MEs using wh-word, head word, direct hypernyms (with depth 6) of head word, unigram, and word shape incrementally. Table 2 shows the question classification accuracy (broken down by question types) of SVM and ME for 6 coarse and 50 fine classes. As can be seen, the main difficulty for question classification lies in the *what* type questions. SVM and ME perform roughly identical if they use the same features. For both SVM and ME, the baseline using the wh-head word and head word results in 81.4% and 82.0% respectively for 50 fine class classification (92.0% and 92.2% for 6 coarse classes). The incremental use of hypernym feature within 6 depths boost about four percent for both SVM and ME under 50 classes, while slight gain or slight loss for SVM and ME for 6 coarse classes. The further use of unigram feature leads to another three percent gain for both SVM and ME in 50 classes. Finally, the use of word shape leads to another 0.6% accuracy increase for both SVM and ME in 50 classes. The best accuracy achieved for 50 classes is 89.2% for SVM and 89.0% for ME. For 6 coarse classes, SVM and ME achieve the best accuracy of 93.4% and 93.6% respectively.

Our best result feature space only consists of 13'697 binary features and each question has 10 to 30 active features. Compared to the over feature size of 200'000 in Li and Roth (2002), our feature space is much more compact, yet turned out to be more informative as suggested by the experiments.

Note that if we replace the bigram with unigram, SVM and ME achieve the overall accuracy of 88.4% and 88.0% respectively for 50 fine classes, and the use of trigram leads SVM and ME to 86.6% and 86.8% respectively. The inclusion of unigram, bigram and trigram together won't boost the accuracy, which reflects the fact that the bigram and trigram features cannot bring more information given that unigram, wh-word and head word features are present. This is because the useful information which are supposed to be captured by bigram or trigram are effectively captured by wh-word and head word features. The unigram feature thus outperforms bigram and trigram due to the fact that it is less sparse. In addition, if we replace the indirect use of hypernym with the direct use of hypernym, the overall accuracy is 84.6% and 84.8% for SVM and ME respectively. All these experiments conform to the individual features contributions as shown in Table 1.

For a better understanding of the error distribution with respect to the 50 question categories, Table 3 shows the precision and recall for each question type in the best result (89.2%) using SVM. It is not surprising that some of the categories are more difficult to predict such as ENTY:other and ENTY:product, while others are much easier such as HUMAN:individual, since the former are more semantically ambiguous than the latter.

Table 3: Precision and recall for fine grained question categories

Class	#	P	R	Class	#	P	R
ABBR	9			desc	7	75.0	85.7
abb	1	100	100	manner	2	100	100
exp	8	88.9	100	reason	6	85.7	100
ENTITY	94			**HUMAN**	65		
animal	16	94.1	100	group	6	71.4	83.3
body	2	100	50.0	individual	55	94.8	100
color	10	100	100	title	1	0.0	0.0
creative	0	100	100	desc	3	100	100
currency	6	100	100	**LOC**	81		
dis.med.	2	40.0	100	city	18	100	77.8
event	2	100	50.0	country	3	100	100
food	4	100	50.0	mountain	3	100	66.7
instrument	1	100	100	other	50	83.9	94.0
lang	2	100	100	state	7	85.7	85.7
letter	0	100	100	**NUM**	113		
other	12	45.5	41.7	code	0	100	100
plant	5	100	100	count	9	81.8	100
product	4	100	25.0	date	47	95.9	100
religion	0	100	100	distance	16	100	62.5
sport	1	100	100	money	3	100	33.3
substance	15	88.9	53.3	order	0	100	100
symbol	0	100	100	other	12	85.7	50.0
technique	1	100	100	period	8	72.7	100
term	7	100	85.7	percent	3	75.0	100
vehicle	4	100	75.0	speed	6	100	83.3
word	0	100	100	temp	5	100	60.0
DESC	138			size	0	100	100
definition	123	89.0	98.4	weight	4	100	75.0

Table 4 shows the summary of the classification accuracy of all models which were applied to UIUC dataset. Note (1) that SNoW accuracy without the related word dictionary was not reported. With the semantically related word dictionary, it achieved 91%. Note (2) that SNoW with a semantically related word dictionary achieved 84.2% but the other algorithms did not use it. Our results are summarized in the last two rows.

Our classifiers are able to classify some chal-

Table 2: Question classification accuracy of SVM and ME using incremental feature sets for 6 and 50 classes

6 coarse classes

Type	#Quest	wh+headword		+headword hypernym		+unigram		+word shape	
		SVM	ME	SVM	ME	SVM	ME	SVM	ME
what	349	88.8	89.1	89.7	88.5	89.7	90.3	90.5	91.1
which	11	90.9	90.9	100	100	100	100	100	100
when	26	100	100	100	100	100	100	100	100
where	27	100	100	100	100	100	100	100	100
who	47	100	100	100	100	100	100	100	100
how	34	100	100	100	100	100	100	100	100
why	4	100	100	100	100	100	100	100	100
rest	2	100	100	50.0	50.0	100	50.0	100	50.0
total	500	92.0	92.2	92.6	91.8	92.8	93.0	93.4	**93.6**

50 fine classes

Type	#Quest	wh+headword		+headword hypernym		+unigram		+word shape	
		SVM	ME	SVM	ME	SVM	ME	SVM	ME
what	349	77.4	77.9	82.8	82.5	85.4	85.1	86.2	86.0
which	11	81.8	90.9	81.8	90.9	90.9	100	90.9	100
when	26	100	100	100	100	100	100	100	100
where	27	92.6	92.6	92.6	92.6	92.6	92.6	92.6	92.6
who	47	100	100	100	100	100	100	100	100
how	34	76.5	76.5	76.5	79.4	97.1	91.2	97.1	91.2
why	4	100	100	100	100	100	100	100	100
rest	2	0.0	0.0	50.0	50.0	0.0	50.0	0.0	50.0
total	500	81.4	82.0	85.4	85.6	88.6	88.4	**89.2**	89.0

lenge questions. For instance, the question *What is the proper name for a female walrus* has been correctly classified as ENTY:animal. However, it still has nearly ten percent error rate for 50 fine classes. The reason is three fold: 1) there are inherently ambiguity in classifying a question. For instance, the question *What is mad cow disease*, it could be either of type DESC:def or ENTY:dismed; 2) there are inconsistent labeling in the training data and test data. For instance, *What is the population of Kansas* is labeled with the type NUM:other while *What is the population of Arcadia , Florida* is labeled with type NUM:count. Another example, *What county is Chicago in* is labeled with type LOC:other while *What county is Phoenix , AZ in* is labeled with type LOC:city; and 3) The parser can produce incorrect parse tree which would result in wrong head word extraction. For instance, the head word extracted from *What is the speed hummingbirds fly* is *hummingbirds* (the correct one should be *speed*), thus leading to the incorrect classification of ENTY:animal (rather than the correct NUM:speed).

5 Conclusion

In contrast to Li and Roth (2006)'s approach which makes use of very rich feature space, we proposed a compact yet effective feature set. In particular, we proposed head word feature and presented two approaches to augment semantic features of such head words using WordNet. In addition, Lesk's word sense disambiguation algorithm was adapted and the depth of hypernym feature was optimized through cross validation, which was to introduce useful information while not bringing too much noise. With further augment of wh-word, unigram feature, and word shape feature, we can obtain accuracy of 89.2% using linear SVMs, or 89.0% using ME for 50 fine classes.

Table 4: Classification accuracy of all models which were applied to UIUC dataset

Algorithm	6 class	50 class
Li and Roth, SNoW	$-^{(1)}$	$78.8^{(2)}$
Hacioglu et al., SVM+ECOC	–	80.2-82
Zhang & Lee, Linear SVM	87.4	79.2
Zhang & Lee, Tree SVM	90.0	–
Krishnan et al., SVM+CRF	93.4	86.2
Linear SVM	93.4	89.2
Maximum Entropy Model	93.6	89.0

6 Acknowledgments

This research was supported by British Telecom grant CT1080028046 and BISC Program of UC Berkeley.

References

Berger, A. L., V. J. D. Pietra, and S. A. D. Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. Computational Linguistics, 22(1):39–71.

Chang, C. C and C. J. Lin. 2001. LIBSVM: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm/.

Charniak, E. and M. Johnson. 2005. Coarse-to-Fine N-Best Parsing and MaxEnt Discriminative Reranking. *The 43rd Annual Meeting on Association for Computational Linguistics*.

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

Fellbaum, C. 1998. An Electronic Lexical Database. The MIT press.

Hacioglu, K. and W. Ward. 2003. Question Classification with Support Vector Machines and Error Correcting Codes. *The Association for Computational Linguistics on Human Language Technology*, vol. 2, pp. 28–30.

Hovy, E., L. Gerber, U. Hermjakob, C. Y. Lin, and D. Ravichandran. 2001. Towards Semantics-based Answer Pinpointing. *The conference on Human language technology research (HLT)*, pp. 1–7.

Khardon, R., D. Roth, and L. G. Valiant. 1999. Relational Learning for NLP using Linear Threshold Elements. *The Conference on Artificial Intelligence*, pp. 911–919.

Klein, D. and C. Manning. 2003. Accurate Unlexicalized Parsing. *The Association for Computational Linguistics*, vol. 1, pp. 423–430.

Krishnan, V., S. Das, and S. Chakrabarti. 2005. Enhanced Answer Type Inference from Questions using Sequential Models. *The conference on Human Language Technology and Empirical Methods in Natural Language Processing*.

Lesk, Michael. 1986. Automatic Sense Disambiguation using Machine Readable Dictionaries: how to tell a pine cone from an ice cream cone. *ACM Special Interest Group for Design of Communication Proceedings of the 5th annual international conference on Systems documentation*, pp. 24–26.

Li, X. and D. Roth. 2002. Learning Question Classifiers. *The 19th international conference on Computational linguistics*, vol. 1, pp. 1–7.

Li, X. and D. Roth. 2006. Learning Question Classifiers: the Role of Semantic Information. *Natural Language Engineering*, 12(3):229–249.

Manning, C. and D. Klein. 2003. Optimization, Maxent Models, and Conditional Estimation without Magic. *Tutorial at HLT-NAACL 2003 and ACL 2003*.

Petrov, S. and D. Klein. 2007. Improved Inference for Unlexicalized Parsing. *HLT-NAACL*.

Pinto, D., M. Branstein, R. Coleman, W. B. Croft, M. King, W. Li, and X. Wei. 2002. QuASM: A System for Question Answering using semi-structured Data *The 2nd ACM/IEEE-CS joint conference on Digital libraries.*.

Radev, D., W. Fan, H. Qi, H. Wu, and A. Grewal. 2002. Probabilistic question answering on the web. *The 11th international conference on World Wide Web*.

Schlaefer, N., J. Ko, J. Betteridge, S. Guido, M. Pathak, and E. Nyberg. 2007. Semantic Extensions of the Ephyra QA System for TREC 2007. *The Sixteenth Text REtrieval Conference (TREC)*.

Seco, N., T. Veale, and J. Hayes. 2004. An Intrinsic Information Content Metric for Semantic Similarity in WordNet. *Proceedings of the European Conference of Artificial Intelligence*.

Vapnik, V. N. 1995. The Nature of Statistical Learning Theory. Springer-Verlag New York.

Voorhees, E. M. and H. T. Dang. 2005. Overview of the TREC 2005 Question Answering Track. *The Text Retrieval Conference (TREC2005)*.

Zhang D. and W. S. Lee. 2003. Question Classification using Support Vector Machines. *The ACM SIGIR conference in informaion retrieval*, pp. 26–32.

CoCQA: Co-Training Over Questions and Answers with an Application to Predicting Question Subjectivity Orientation

Baoli Li
Emory University
csblli@gmail.com

Yandong Liu
Emory University
yliu49@emory.edu

Eugene Agichtein
Emory University
eugene@mathcs.emory.edu

Abstract

An increasingly popular method for finding information online is via the Community Question Answering (CQA) portals such as Yahoo! Answers, Naver, and Baidu Knows. Searching the CQA archives, and ranking, filtering, and evaluating the submitted answers requires intelligent processing of the questions and answers posed by the users. One important task is automatically detecting the question's *subjectivity orientation*: namely, whether a user is searching for subjective or objective information. Unfortunately, real user questions are often vague, ill-posed, poorly stated. Furthermore, there has been little labeled training data available for real user questions. To address these problems, we present *CoCQA*, a co-training system that exploits the association between the questions and contributed answers for question analysis tasks. The co-training approach allows *CoCQA* to use the effectively unlimited amounts of *unlabeled* data readily available in CQA archives. In this paper we study the effectiveness of *CoCQA* for the question subjectivity classification task by experimenting over thousands of real users' questions.

1 Introduction

Automatic question answering (QA) has been one of the long-standing goals of natural language processing, information retrieval, and artificial intelligence research. For a natural language question we would like to respond with a specific, accurate, and complete answer that addresses the question. Although much progress has been made, answering complex, opinion, and even many factual questions automatically is still beyond the current state-of-the-art. At the same time, the rise of popularity in social media and collaborative content creation services provides a promising alternative to web search or completely automated QA. The explicit support for social interactions between participants, such as posting comments, rating content, and responding to questions and comments makes this medium particularly amenable to Question Answering. Some very successful examples of Community Question Answering (CQA) sites are Yahoo! Answers[1] and Naver[2], and Baidu Knows[3]. Yahoo! Answers alone has already amassed hundreds of millions of answers posted by millions of participants on thousands of topics.

The questions posted to such CQA portals are typically complex, subjective, and rely on human interpretation to understand the corresponding information need. At the same time, the questions are also usually ill-phrased, vague, and often *subjective* in nature. Hence, analysis of the questions (and of the corresponding user intent) in this setting is a particularly difficult task. At the same time, CQA content incorporates the relationships between questions and the corresponding answers. Because of the various incentives provided by the CQA sites, answers posted by users tend to be, at least to some degree, responsive to the question. This observation suggests investigating whether the relation-

[1] http://answers.yahoo.com
[2] http://www.naver.com
[3] http://www.baidu.com

ship between questions and answers can be exploited to improve automated analysis of the CQA content and the user intent behind the questions posted.

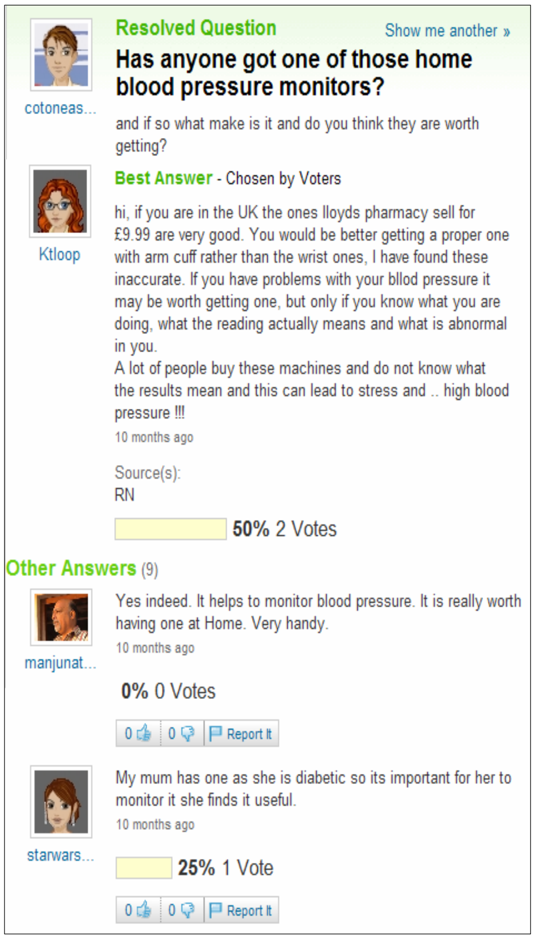
To this end, we exploit the ideas of *co-training*, a general semi-supervised learning approach naturally applicable to cases of complementary views on a domain, for example, web page links and content (Blum and Mitchell, 1998). In our setting, we focus on the complimentary views for a question, namely the text of the question and the text of the associated answers.

As a concrete case-study of our approach we focus on one particularly important aspect of intent detection: the *subjectivity orientation*. We attempt to predict whether a question posted in a CQA site is subjective or objective. Objective questions are expected to be answered with reliable or authoritative information, typically published online and possibly referenced as part of the answer, whereas subjective questions seek answers containing private states, e.g. personal opinions, judgment, experiences. If we could automatically predict the orientation of a question, we would be able to better rank or filter the answers, improve search over the archives, and more accurately identify similar questions. For example, if a question is objective, we could try to find a few highly relevant articles as references, whereas if a question is subjective, useful answers are not expected to be found in authoritative sources and tend to rank low with current question answering and CQA search techniques. Finally, learning how to identify question orientation is a crucial component of inferring user intent, a long-standing problem in web information access settings.

In particular, we focus on the following research questions:

- Can we utilize the inherent structure of the CQA interactions and use the unlimited amounts of *unlabeled* data to improve classification performance, and/or reduce the amount of manual labeling required?

- Can we automatically predict question subjectivity in Community Question Answering – and which features are useful for this task in the real CQA setting?



Resolved Question

Show me another »

cotoneas...

Has anyone got one of those home blood pressure monitors?

and if so what make is it and do you think they are worth getting?

Ktloop

Best Answer - Chosen by Voters

hi, if you are in the UK the ones lloyds pharmacy sell for £9.99 are very good. You would be better getting a proper one with arm cuff rather than the wrist ones, I have found these inaccurate. If you have problems with your bllod pressure it may be worth getting one, but only if you know what you are doing, what the reading actually means and what is abnormal in you.
A lot of people buy these machines and do not know what the results mean and this can lead to stress and .. high blood pressure !!!

10 months ago

Source(s):
RN

50% 2 Votes

Other Answers (9)

manjunat...

Yes indeed. It helps to monitor blood pressure. It is really worth having one at Home. Very handy.

10 months ago

0% 0 Votes

0 0 Report It

starwars...

My mum has one as she is diabetic so its important for her to monitor it she finds it useful.

10 months ago

25% 1 Vote

0 0 Report It

Figure 1: Example question (Yahoo! Answers)

The rest of the paper is structured as follows. We first overview the community question answering setting, and state the question orientation classification problem, which we use as the motivating application for our system, more precisely. We then introduce our *CoCQA* system for semi-supervised classification of questions and answers in CQA communities (Section 3). We report the results of our experiments over thousands of real user questions in Section 4, showing the effectiveness of our approach. Finally, we review related work in Section 5, and discuss our conclusions and future work in Section 6.

2 Question Orientation in CQA

We first briefly describe the essential features of question answering communities such as Yahoo! Answers or Naver. Then, we formally state the problem addressed in this paper, and the features used for this setting.

2.1 Community Question Answering

Online social media content and associated services comprise one of the fastest growing segments on the Web. The explicit support for social interactions between participants, such as posting comments, rating content, and responding to questions and comments makes the social media unique. Question answering has been particularly amenable to social media by directly connecting information seekers with the community members willing to share the information. Yahoo! Answers, with millions of users and hundreds of millions of answers for millions of questions is a very successful implementation of CQA.

For example, consider two example user-contributed questions, objective and subjective respectively:

> **`Q1: What's the difference between chemotherapy and radiation treatments?`**
>
> **`Q2: Has anyone got one of those home blood pressure monitors?`** `and if so what make is it` *`and do you think they are worth getting`*`?`

Figure 1 shows an example of community interactions in Yahoo! Answers around the question Q2 above. A user posted the question in the *Health* category of the site, and was able to obtain 10 responses from other users. Eventually, the asker chooses the best answer. Failing that, as shown in the example, the best answer can also be chosen according to the votes from other users. Many of the interactions depend on the perceived goals of the asker: if the participants interpret the question as subjective, they will tend to share their experiences and opinions, and if they interpret the question as objective, they may still share their experiences but may also provide more factual information.

2.2 Problem Definition

We now state our problem of question orientation more precisely. We consider question orientation from the perspective of user goals: authors of objective questions request authoritative, objective information (e.g., published literature or expert opinion), whereas authors of subjective questions seek opinions or judgments of other users in the community. We state our problem as follows.

> **Question Subjectivity Problem:** *Given a question Q in a question answering community, predict whether Q has objective or subjective orientation, based on question and answer text as well as the user and community feedback.*

3 *CoCQA*: A Co-Training Framework over Questions and Answers

In the CQA setting we could easily obtain thousands or millions of unlabeled examples from the online CQA archives. On the other hand, it is difficult to create a labeled dataset with a reasonable size, which could be used to train a perfect classifier to analyze questions from different domains and sub-domains. Therefore, semi-supervised learning (Chapelle et al., 2006) is a natural approach for this setting.

Intuitively, we can consider the text of the question itself or answers to it. In other words, we have multiple (at least two) natural views of the data, which satisfies the conditions of the co-training approach (Blum and Mitchell, 1998). In *co-training*, two separate classifiers are trained on two sets of features, respectively. By automatically labeling the unlabeled examples, these two classifiers iteratively "teach" each other by giving their partners a newly labeled data that they can predict with high confidence. Based on the original co-training algorithm in (Blum and Mitchell, 1998) and other implementations, we develop our algorithm *CoCQA* shown in Figure 2.

At Steps 1 and 2, the *K* examples are coming from different feature spaces, and each category (for example, *Subjective* and *Objective*) has top K_j most confident examples chosen, where K_j corresponds to the distribution of class in the current set of labeled examples L. *CoCQA* will terminate when the increments of both classifiers are less than a specified threshold *X* or the maximum number of iterations are exceeded. Following the co-training approach, we include the most confidently predicted examples as additional "labeled" data. The SVM output margin value was used to estimate confidence; alternative

```
Input:
  • $F_Q$ and $F_A$ are Question and Answer feature views
  • $C_Q$ and $C_A$ are classifiers trained on $F_Q$ and $F_A$ respectively
  • L is a set of labeled training examples
  • U is a set of unlabeled examples
  • $K$: Number of unlabeled examples to choose on each iteration
  • $X$: the threshold for increment
  • $R$: the maximal number of iterations
Algorithm CoCQA
1. Train $C_Q,0$ on L: $F_Q$, and record resulting $ACC_Q,0$
2. Train $C_A,0$ on L: $F_A$, and record resulting $ACC_A,0$
3. for j=1 to $R$ do:
     Use $C_Q,j-1$ to predict labels for U and choose
         top $K$ items with highest confidence $\rightarrow E_{Q,},j-1$
     Use $C_A,j-1$ to predict labels for U and choose
         top $K$ items with highest confidence $\rightarrow E_{A,},j-1$
     Move examples $E_{Q,},j-1 \cup E_{A,},j-1 \rightarrow$ L
     Train $C_Q,j$ on L: $F_Q$ and record training $ACC_Q,j$
     Train $C_A,j$ on L: $F_A$ and record training $ACC_A,j$
     if $Max(\Delta ACC_Q,j, \Delta ACC_A,j) < X$ break

4. return final classifiers $C_Q,j \rightarrow C_Q$ and $C_A,j \rightarrow C_A$
```

Figure 2: Algorithm CoCQA: A co-training algorithm for exploiting redundant feature sets in community question answering.

methods (including reliability of this confidence prediction) could further improve performance, and we will explore these issues in future work. Finally, the next question is how to estimate classification performance with training data. For each pass, we randomly split the original training data into *N* folds (N=10 in our experiments), and keep one part for validation and the rest, augmented with the newly added examples, as the expanded training set.

After *CoCQA* terminates, we obtain two classifiers. When a new example arrives, we will classify it with these two classifiers based on both of the feature sets, and combine the predictions of these two classifiers. We explored two strategies to make the final decision based on the confidence values given by two classifiers:

- Choose the class with higher confidence
- Multiply the confidence values, and choose the class that has the highest product.

We found the second heuristic to be more effective than the first in our experiments. As the base classifier we use SVM in the current implementation, but other classifiers could be incorporated as well.

4 Experimental Evaluation

We experiment with supervised and semi-supervised methods on a relatively large data set from Yahoo! Answers.

4.1 Datasets

To our knowledge, there is no standard dataset of real questions and answers posted by online users, labeled for subjectivity orientation. Hence, we had to create a dataset ourselves. To create our dataset, we downloaded more than 30,000 resolved questions from each of the following top-level categories of Yahoo! Answers: Arts, Education, Health, Science, and Sports. We randomly chose 200 questions from each category to create a raw dataset with 1,000 questions total. Then, we labeled the dataset with annotators from the Amazon's Mechanical Turk service[4].

For annotation, each question was judged by 5 Mechanical Turk workers who passed a qualification test of 10 questions (labeled by ourselves) with at least 9 of them correctly marked. The qualification test was required to ensure that the raters were sufficiently competent to make reasonable judgments. We grouped the tasks into 25 question batches, where the whole batch was submitted as the Mechanical Turk's Human Intelligence Task (HIT). The batching of questions was done to easily detect the "random" ratings produced by irresponsible workers. That is, each worker rated a batch of 25 questions.

While precise definition of subjectivity is elusive, we decided to take the practical perspective, namely the "majority" interpretation. The annotators were instructed to guess orientation according to how the question would be answered by most people. We did not deal with multi-part questions: if any part of question was subjective, the whole question was labeled as subjective. The gold standard was thus derived with the majority strategy, followed by manual inspection as a "sanity check". At this stage we removed 22 questions with undeterminable meaning, including gems such as "`Upward Soccer`

[4] http://www.mturk.com

`Shorts?`"[5] and "`1+1=?fdgdgdfg?`"[6]. Finally, we create a labeled dataset consisting of 978 resolved questions, available online[7].

	Num. of SUB. Q	*Num. of OBJ. Q*	*Total Num.*	*Annotator agreement*
Arts	137 (70%)	58 (30%)	195	0.841
Education	127 (64%)	70 (36%)	197	0.716
Health	125 (64%)	69 (36%)	194	0.833
Science	103 (52%)	94 (48%)	197	0.618
Sports	154 (79%)	41 (21%)	195	0.877
Total	**646 (66%)**	**332 (34%)**	**978**	**0.777**

Table 1: Labeled dataset statistics.

Table 1 reports the statistics of the annotated dataset. The overall inter-annotator percentage agreement between Mechanical Turk workers and final annotation is 0.777, indicating that the task is difficult, but feasible for humans to annotate manually.

The statistics of our labeled sample show that the vast majority of the questions in all categories except for Science are subjective in nature. The relatively high ratio of subjective questions in the Science category is surprising. However, we find that users often post polemics and statements instead of questions, using CQA as a forum to share their opinions on controversial topics. Overall, we were struck by the expressed need in Subjective information, even for categories such as Health and Education, where objective information would intuitively seem more desirable.

4.2 Features Used in Experiments

For the subjectivied experiments to follow, we attempt to capture the linguistic characteristics identified in previous work (Section 5) in a lightweight and robust manner, due to the informal and noisy nature of CQA. In particular, we use the following feature classes, computed separately over question and answer content:

- Character 3-grams
- Words
- Word with Character 3-grams
- Word n-grams (n<=3, i.e. W_i, W_iW_{i+1}, $W_iW_{i+1}W_{i+2}$)
- Word and POS n-gram (n<=3, i.e. W_i, W_iW_{i+1}, W_i POS_{i+1}, POS_iW_{i+1}, POS_iPOS_{i+1}, etc.).

We use the character 3-grams features to overcome spelling errors and problems of ill-formatted or ungrammatical questions, and the POS information to capture common patterns across domains, as words, especially the content words, are quite diverse in different topical domains. For word and character 3-gram features, we consider two different versions: case-sensitive and case-insensitive. Case-insensitive features are assumed to be helpful for mitigating negative effects of ill-formatted text.

Moreover, we experimented with three term weighting schemes: Binary, TF, and TF*IDF. Term frequency (TF) exhibited better performance in our development experiments, so we use this weighting scheme for all the experiments in Section 4. Regarding features: both words and structure of the text (e.g., word order) can be used to infer subjectivity. Therefore, the features we employ, such as words and word n-grams, are expected to be useful as a (coarse) proxy to grammatical and phrase features. Unlike traditional work on news-like text, the text of CQA and has poor spelling, grammar, and heavily uses non-standard abbreviations, hence our decision to use character n-grams.

4.3 Experimental Setting

Metrics: Since the prediction on both subjective questions and objective questions is equally important, we use the **macro-averaged F1** measure as the evaluation metric. This is computed as the macro average of F1 measures computed for the *Subjective* and *Objective* classes individually. The **F1** measure for either class is computed as $\frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$.

Methods compared: We compare our approach with both the base supervised learning, as well as *GE*, a state-of-the-art semi-supervised method:

- ***Supervised***: we use the LibSVM implementation (Chang and Lin, 2001) with linear kernel.

[5]http://answers.yahoo.com/question/?qid=20060829074901AADBRJ4
[6] http://answers.yahoo.com/question/?qid=1006012003651
[7] Available at http://ir.mathcs.emory.edu/datasets/.

- ***GE***: This is a state-of-the-art semi-supervised learning algorithm, Generalized Expectation (GE), introduced in (McCallum et al., 2007) that incorporates model expectations into the objective functions for parameter estimation.
- ***CoCQA***: Our method (Section 3).

For semi-supervised learning experiments, we selected a random subset of 2,000 unlabeled questions for each of the topical categories, for the total of 10,000 unlabeled questions.

4.4 Experimental Results

First we report the performance of our *Supervised* baseline system with a variety of features, reporting the average results of 5-fold cross validation. Then we investigate the performance to our new *CoCQA* framework under a variety of settings.

4.4.1 Supervised Learning

Table 2 reports the classification performance for varying units of representation (e.g., question text vs. answer text) and the varying feature sets. We used case-insensitive features and TF (term frequency within the text unit) as feature weights, as these two settings achieved the best results in our development experiments. The rows show performance considering only the question text (**question)**, the best answer (**best_ans**), text of all answers to a question (**all_ans**), the text of the question and the best answer (**q_bestans**), and the text of the question with all answers (**q_allans**), respectively. In particular, using the words in the question alone achieves F1 of 0.717, compared to using words in the question and the best answers text (F1 of 0.695). For comparison, a naïve baseline that always guesses the majority class (*Subjective*) obtains F1 of 0.398.

With character 3-gram, our system achieves performance comparable with words as features, but combining them together does not improve performance. We observe a slight gain with more complicated features, e.g. word n-gram, or word and POS n-grams, but the gain is not significant, and hence not worth the increased complexity of the feature generation. Finally, combining question text with answer text does not improve performance.

Interestingly, the best answer itself is not as effective as the question for subjectivity analysis, nor is using all of the answers submitted. One possible reason is that approximately 40% of the best answers were chosen by the community and not the asker herself, are hence not necessarily representative of the asker intent.

Feature set / Unit	Char 3-gram	Word	Word+ Char 3-gram	Word n-gram (n<=3)	Word POS n-gram (n<=3)
question	0.700	**0.717**	0.694	0.716	0.720
best_ans	0.587	0.597	0.578	0.580	0.565
all_ans	0.603	0.628	0.607	0.648	0.630
q_bestans	0.681	**0.695**	0.662	0.687	0.712
q_allans	0.679	0.677	0.676	0.708	0.689
Naïve (majority class) baseline:			0.398		

Table 2. Performance of predicting question orientation on the mixed dataset with varying feature sets (Supervised).

Table 3 reports the supervised subjectivity classification performance for each question category with word features. The overall classification results are significantly lower compared to training and testing on the mixture of the questions drawn from all categories, likely caused by the small amount of labeled training data for each category. Another possibility is that the subjectivity clues are not topical and hence are not category dependent, with the possible exception of the questions in the Health domain.

Category	Arts	Edu.	Health	Science	Sports
F1	0.448	0.572	**0.711**	0.647	0.441

Table 3. Experiment results on sub-categories with supervised SVM (q_bestans features).

As words are simple and effective features in this experiment, we will use them in the subsequent experiments. Furthermore, the feature set using the words in the question with best answer together (**q_bestans**) exhibit higher performance than question with all answers (**q_allans**). Thus, we will only consider questions and best answers in the following experiments with GE and *CoCQA*.

4.4.2 Semi-Supervised Learning

We now focus on investigating the effectiveness of *CoCQA*, our co-training-based framework for community question answering analysis. Table 4 summarizes the main results of this section. The values for *CoCQA* are derived with the parameter settings: K=100, *X*=0.001. These optimal settings are chosen after comprehensive experiments with different combinations, described later in this section. *GE* does not exhibit a significant improvement over *Supervised*. In contrast, *CoCQA* performs significantly better than the purely supervised method, with F1 of 0.745 compared to the F1 of 0.717 for *Supervised*. While it may seem surprising that a semi-supervised method outperforms a supervised one, note that we use all of the available *labeled* data as provided to the Supervised method, as well as a large amount of *unlabeled* data, that is ultimately responsible for the performance improvement.

Features / Method	Question	Question+ Best Answer
Supervised	0.717	0.695
GE	0.712 (-0.7%)	0.717 (+3.2%)
CoCQA	0.731 (+1.9%)	**0.745** (+7.2%)

Table 4. Performance of *CoCQA*, *GE*, and *Supervised* with the same feature and data settings.

As an added advantage, *CoCQA* approach is also practical. In a realistic application, we have two different situations: offline and online. With online processing, we may not have best answers available to predict question's orientation, whereas we can employ information from best answers in offline setting. Co-training is a solution that is applicable to both situations. With *CoCQA*, we have two classifiers using the question text and the best answer text, respectively. We can use both of them to obtain better results in the offline setting, while in online setting, we can use the text of the question alone. In contrast, *GE* may not have this flexibility.

We now analyze the performance of *CoCQA* under a variety of settings to derive optimal parameters and to better understand the performance. Figure 3 reports the performance of *CoCQA* with varying the *K* parameter from 20 to 200. In this experiment, we fix *X* to be 0.001. The combination of question and best answer is superior to that of question and all answers. When *K* is 100, the system obtains the best result, 0.745.

Figure 4 reports the number of co-training iterations needed to converge to optimal performance. After 13 iterations (and 2500 unlabeled examples added), *CoCQA* achieves optimal performance, and eventually terminates after an additional 3 iterations. While a validation set should have been used for *CoCQA* parameter tuning, Figures 3 and 4 indicate that *CoCQA* is not sensitive to the specific parameter settings. In particular, we observe that any *K* is greater than 100, and for any number of iterations *R* is greater than 10, *CoCQA* exhibits in effectively equivalent performance.

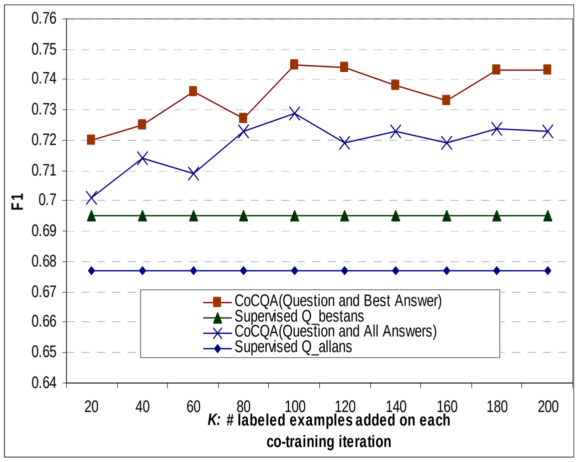


Figure 3: Performance of CoCQA for varying the *K* (number of examples added on each iteration of co-training).

Figure 5 reports the performance of *CoCQA* for varying the number of labeled examples from 50 to 400 (that is, up to 50% of the available labeled training data). Note that for this comparison we use the same feature sets (words in question and best answer text), but using only the (varying) fractions of the manually labeled data. Surprisingly, *CoCQA* exhibits comparable performance of F1=0.685 with only 200 labeled examples are used, compared to the F1=0.695 for *Supervised* with all 800 labeled training examples on this feature set. In other words, *CoCQA* is able to achieve comparable performance to supervised SVM with only one quarter of the labeled training data.

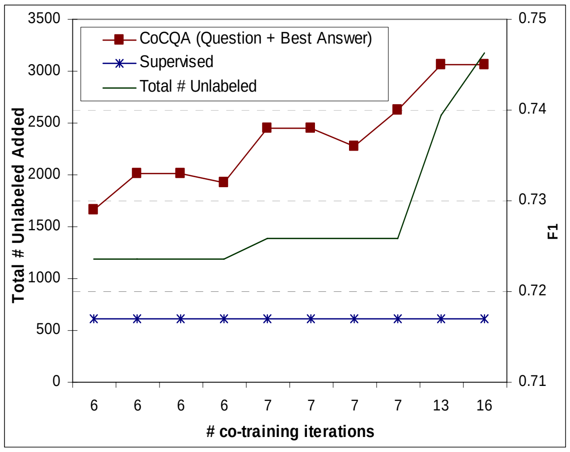


Figure 4: Performance and the *total* number of unlabeled examples added for varying number of co-training iterations (*K*=100, using *q_bestans* features)

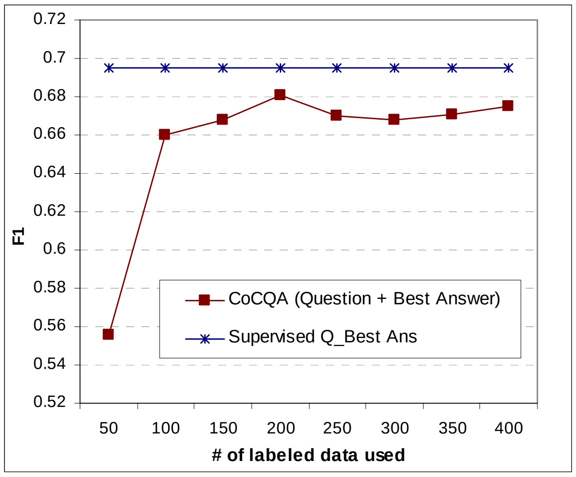


Figure 5: Performance of *CoCQA* with varying number of labeled examples used, compared to *Supervised* method, on same features.

5 Related Work

Question analysis, especially question classification, has been long studied in the question answering research community. However, most of the previous research primarily considered factual questions, with the notable exception of the most recent TREC opinion QA track. Furthermore, the questions were specifically designed for benchmark evaluation. A related thread of research considered deep analysis of the questions (and corresponding sentences) by manually classifying questions along several orientation dimensions, notably (Stoyanov et al., 2005). In contrast, our work focuses on analyzing real user questions posted in a question answering community. These questions are often complex or subjective, and are typically difficult to answer automatically as the question author probably was not able to find satisfactory answers with quick web search.

Automatic complex question answering has been an active area of research, ranging from simple modification to factoid QA techniques (e.g., Soricut and Brill, 2003) to knowledge intensive approaches for specific domains (e.g., Harabagiu et al. 2001, Fushman and Lin 2007). However, the technology does not yet exist to automatically answer open-domain complex and subjective questions. While there has been some recent research (e.g., Agichtein et al. 2008, Bian et al. 2008) on retrieving high quality answers from CQA archives, the subjectivity orientation of the questions has not been considered as a feature for ranking.

A related corresponding problem is complex QA evaluation. Recent efforts at automatic evaluation show that even for well-defined, objective, complex questions, evaluation is extremely labor-intensive and introduces many challenges (Lin and Fushman 2006, Lin and Zhang 2007). As part of our contribution we showed that it is feasible to use the Amazon Mechanical Turk service for evaluation by combining large degree of annotator redundancy (5 annotators per question) with more sparse but higher-quality expert annotation.

The problem of automatic subjective question answering has recently started to be addressed in the question answering community, most recently as the first opinion QA track in (Dang et al., 2007). Unlike the controlled TREC opinion track (introduced in 2007), many of the questions in Yahoo! Answers community are inherently subjective, complex, ill-formed, or all of the above. To our knowledge, this paper is the first large-scale study of subjective/objective orientation of information needs, and certainly the first in the CQA environment.

A closely related research thread is subjectivity analysis at document and sentence level. For example, reference (Yu, H., and Hatzivassiloglou, V. 2003; Somasundaran et al. 2007) attempted to classify sentences into those reporting facts or opinions. Also related is research on sentiment analysis (e.g., Pang et al., 2004) where the goal is to classify a sentence or text fragment as being overall positive or negative. More generally, (Wiebe et al. 2004) and subsequent work focused on the analysis of subjective language in narrative text, primarily news. Our problem is quite different in the sense that we are trying to identify the orientation of a *question*. Nevertheless, our baseline method is similar to the methods and features used for sentiment analysis, and one of our contributions is evaluating the usefulness of the established features and techniques to the new CQA setting.

In order to predict question orientation, we build on co-training, one of the known semi-supervised learning techniques. Many models and techniques have been proposed for classification, including support vector machines, decision tree based techniques, boosting-based techniques, and many others. We use LIBSVM (Chang and Lin, 2001) as a robust implementation of SVM algorithms.

In summary, while we draw on many techniques in question answering, natural language processing, and text classification, our work differs from previous research in that a) develop a novel co-training based algorithm for question and answer classification; b) we address a relatively new problem of *automatic* question subjectivity prediction; c) demonstrate the effectiveness of our techniques in the new CQA setting and d) explore the characteristics unique to CQA – while showing good results for a quite difficult task.

6 Conclusions

We presented *CoCQA*, a co-training framework for modeling the textual interactions in question answer communities. Unlike previous work, we have focused on real user questions (often noisy, ungrammatical, and vague) submitted in Yahoo! Answers, a popular community question answering portal. We demonstrated *CoCQA* for one particularly important task of automatically identifying question subjectivity orientation, showing that *CoCQA* is able to exploit the structure of questions and corresponding answers. Despite the inherent difficulties of subjectivity analysis for real user questions, we have shown that by applying *CoCQA* to this task we can significantly improve prediction performance, and substantially reduce the size of the required training data, while outperforming a general state-of-the-art semi-supervised algorithm that does not take advantage of the CQA characteristics.

In the future we plan to explore more sophisticated features such semantic concepts and relationships (e.g., derived from WordNet or Wikipedia), and richer syntactic and linguistic information. We also plan to explore related variants of semi-supervised learning such as co-boosting methods to further improve classification performance. We will also investigate other applications of our co-training framework to tasks such as sentiment analysis in community question answering and similar social media content.

Acknowledgments

This research was partially supported by the Emory University Research Committee (URC) grant, and by the Emory College Seed grant. We thank the Yahoo! Answers team for providing access to the Answers API, and anonymous reviewers for their excellent suggestions.

References

Agichtein, E., Castillo, C., Donato, D., Gionis, A., and Mishne, G. 2008. Finding High-Quality Content in Social Media with an Application to Community-Based Question Answering. *WSDM2008*

Bian, J., Liu, Y., Agichtein, E., and H. Zha. 2008, to appear. Finding the Right Facts in the Crowd: Factoid Question Answering over Social Media, *Proceedings of the Inter-national World Wide Web Conference (WWW), 2008*

Blum, A., and Mitchell, T. 1998. Combining Labeled and Unlabeled Data with Co-Training. *Proc. of the Annual Conference on Computational Learning Theory*.

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

Chapelle, O., Scholkopf, B., and Zien, A. 2006. *Semi-supervised Learning*. The MIT Press, Cambridge, Mas-sachusetts.

Dang, H. T., Kelly, D., and Lin, J. 2007. Overview of the TREC 2007 Question Answering track. *In Proceedings of TREC-2007.*

Demner-Fushman, D. and Lin, J. 2007. Answering clinical questions with knowledge-based and statistical techniques. *Computational Linguistics*, 33(1):63–103.

Harabagiu, S., Moldovan, D., Pasca, M., Surdeanu, M. , Mihalcea, R., Girju, R., Rusa, V., Lacatusu, F., Morarescu, P., and Bunescu, R. 2001. Answering Complex, List and Context Questions with LCC's Question-Answering Server. *In Proc. of TREC 2001*.

Lin, J. and Demner-Fushman, D. 2006. Methods for automatically evaluating answers to complex questions. *In-formation Retrieval*, 9(5):565–587

Lin, J. and Zhang, P. 2007. Deconstructing nuggets: the stability and reliability of complex question answering evaluation. *In Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, pages 327–334.*

Mann, G., and McCallum, A. 2007. Simple, Robust, Scalable Semi-supervised Learning via Expectation Regularization. *Proceedings of ICML 2007*.

Pang, B., and Lee, L. 2004. A Sentimental Education: Sen-timent Analysis Using Subjective Summarization Based on Minimum Cuts. *In Proc. of ACL.*

Prager, J. 2006. Open-Domain Question-Answering. *Foundations and Trends in Information Retrieval*.

Sindhwani, V., Keerthi, S. 2006. Large Scale Semi-supervised Linear SVMs. *Proceedings of SIGIR 2006*.

Somasundaran, S., Wilson, T., Wiebe, J. and Stoyanov, V. 2007. QA with Attitude: Exploiting Opinion Type Analysis for Improving Question Answering in On-line Discussions and the News. *In proceedings of International Conference on Weblogs and Social Media (ICWSM-2007)*.

Soricut, R. and Brill, E. 2004. Automatic question answering: Beyond the factoid. *Proceedings of HLT-NAACL.*

Stoyanov, V., Cardie, C., and Wiebe, J. 2005. Multi-Perspective question answering using the OpQA corpus. In Proceedings of EMNLP.

Tri, N. T., Le, N. M., and Shimazu, A. 2006. Using Semi-supervised Learning for Question Classification. *In Proceedings of ICCPOL-2006*.

Wiebe, J., Wilson, T., Bruce R., Bell M., and Martin M. 2004. Learning subjective language. *Computational Linguistics*, 30 (3).

Yu, H., and Hatzivassiloglou, V. 2003. Towards Answering Opinion Questions: Separating Facts from Opinions and Identifying the Polarity of Opinion Sentences. *In Proceedings of EMNLP-2003.*

Zhang, D., and Lee, W.S. 2003. Question Classification Using Support Vector Machines. *Proceedings of the 26th Annual International ACM SIGIR Conference on Re-search and Development in Information Retrieval*.

Zhu, X. 2005. Semi-supervised Learning Literature Survey. *Technical Report 1530*, Computer Sciences, University of Wisconsin-Madison.

Automatic Set Expansion for List Question Answering

Richard C. Wang, Nico Schlaefer, William W. Cohen, and Eric Nyberg
Language Technologies Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213
{rcwang,nico,wcohen,ehn}@cs.cmu.edu

Abstract

This paper explores the use of set expansion (SE) to improve question answering (QA) when the expected answer is a list of entities belonging to a certain class. Given a small set of seeds, SE algorithms mine textual resources to produce an extended list including additional members of the class represented by the seeds. We explore the hypothesis that a noise-resistant SE algorithm can be used to extend candidate answers produced by a QA system and generate a new list of answers that is better than the original list produced by the QA system. We further introduce a hybrid approach which combines the original answers from the QA system with the output from the SE algorithm. Experimental results for several state-of-the-art QA systems show that the hybrid system performs better than the QA systems alone when tested on list question data from past TREC evaluations.

1 Introduction

Question answering (QA) systems are designed to retrieve precise answers to questions posed in natural language. A *list question* expects a list as its answer, e.g. *Name the coffee-producing countries in South America*. The ability to answer list questions has been tested as part of the yearly TREC QA evaluation (Dang et al., 2006; Dang et al., 2007). This paper focuses on the use of *set expansion* to improve list question answering. A set expansion (SE) algorithm receives as input a few members of a class or set, and mines various textual resources (e.g. web pages) to produce an extended list including additional members of the class or set that are not in the input. A well-known online SE system is Google Sets[1]. This system is publicly accessible, but since it is a proprietary system that might be changed at any time, its results cannot be replicated reliably. We explore the hypothesis that a SE algorithm, when carefully designed to handle noisy inputs, can be applied to the output from a QA system to produce an overall list of answers for a given question that is better than the answers produced by the QA system itself. We propose to exploit large, redundant sources of structured and/or semi-structured data and use linguistic analysis to seed a shallow analysis of these sources. This is a hard problem since the linguistic evidence used as seeds is noisy. More precisely, we combine the QA system Ephyra (Schlaefer et al., 2007) with the SE system SEAL (Wang and Cohen, 2007) to create a hybrid approach that performs better than either system by itself when tested on data from the TREC 13-15 evaluations. In addition, we apply our SE algorithm to answers generated by the five QA systems that performed the best on the list questions in the TREC 15 evaluation and report improvements in F_1 scores for four of these systems.

Section 2 of the paper gives an overview of the QA and SE systems used for our experiments. Section 3 describes how the SE system was adapted to deal with noisy seeds produced by QA systems, and Section 4 presents the details of the experimental design. Experimental results are discussed in Section 5, and the paper concludes in Section 6 with a discussion of planned future work.

[1]http://labs.google.com/sets

2 System Overview

2.1 Ephyra Question Answering System

Ephyra (Schlaefer et al., 2006; Schlaefer et al., 2007) is a QA system that has been evaluated in the TREC QA track (Dang et al., 2006; Dang et al., 2007). The system combines three answer extraction techniques for factoid and list questions: (1) an answer type classification approach; (2) a syntactic pattern learning and matching approach; and (3) a semantic extractor that uses a semantic role labeling system. The answer type based extractor classifies questions by their answer types and extracts candidates of the expected types. The Ephyra pattern matching approach learns textual patterns that relate question key terms to possible answers and applies these patterns to candidate sentences to extract factoid answers. The semantic approach generates a semantic representation of the question that is based on predicate-argument structures and extracts answer candidates from similar structures in the corpus. The source code of the answer extractors is included in OpenEphyra, an open source release of the system.[2]

The answer candidates from these extractors are combined and ranked by a statistical answer selection framework (Ko et al., 2007), which estimates the probability of an answer based on a number of answer validation and similarity features. Validation features use resources such as gazetteers and Wikipedia to verify an answer, whereas similarity features measure the syntactic and semantic similarity to other candidates, e.g. using string distance measures and WordNet relations.

2.2 Set Expander for Any Language (SEAL)

Set expansion (SE) refers to expanding a given partial set of objects into a more complete set. SEAL[3] (Wang and Cohen, 2007) is a SE system which accepts input elements (seeds) of some target set S_t and automatically finds other probable elements of S_t in semi-structured documents such as web pages. SEAL also works on unstructured text, but its extraction mechanism benefits from structuring elements such as HTML tags. The algorithm is independent of the human language from which the seeds are taken, and also independent of the markup language used to annotate the documents. Examples of SEAL's input and output are shown in Figure 1.

	English	Chinese	Japanese
input seed	Amazing Race	豬血糕	ドラえもん
	Survivor	臭豆腐	ハローキティ
output	Big Brother	蘿蔔糕	アンパンマン
	The Mole	蚵仔煎	シナモロール
	The Apprentice	肉圓	ウルトラマン
	Project Runway	滷肉飯	スヌーピー
	The Bachelor	春捲	くまのプーさん

Figure 1: Examples of SEAL's input and output. English entities are reality TV shows, Chinese entities are popular Taiwanese food, and Japanese entities are famous cartoon characters.

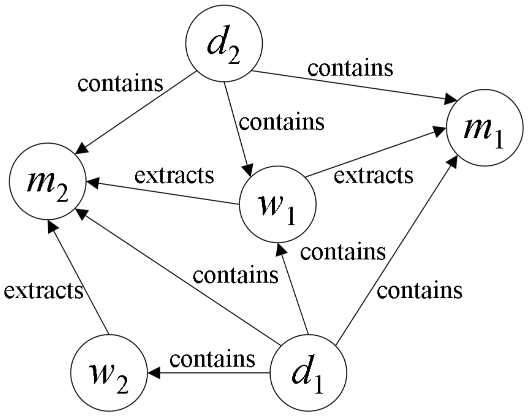


Figure 2: An example graph constructed by SEAL. Every edge from node x to y actually has an inverse relation edge from node y to x that is not shown here (e.g. m_1 *is extracted by* w_1).

In more detail, SEAL comprises three major components: the Fetcher, the Extractor, and the Ranker. The *Fetcher* focuses on retrieving web pages. The URLs of the web pages come from top results retrieved from Google and Yahoo! using the concatenation of all seeds as the query. The *Extractor* automatically constructs page-specific extraction rules, or *wrappers*, for each page that contains the seeds. Every wrapper is defined by two character strings, which specify the left-context and right-context necessary for an entity to be extracted from a page. These strings are chosen to be maximally-long contexts that bracket at least one occurrence of every seed string on a page. Most of the wrappers con-

[2]http://www.ephyra.info/

[3]http://rcwang.com/seal

tain HTML tags, which illustrates the importance of structuring information in the source documents. All entity mentions bracketed by these contextual strings derived from a particular page are extracted from the same page. Finally, the *Ranker* builds a graph, and then ranks the extracted mentions globally based on the weights computed by performing a random graph walk.

An example graph is shown in Figure 2, where each node d_i represents a document, w_i a wrapper, and m_i an extracted entity mention. The graph models the relationship between documents, wrappers, and mentions. In order to measure the relative importance of each node within the graph, the Ranker performs a graph walk until all node weights converge. The idea is that nodes are weighted higher if they are connected to many other highly weighted nodes.

We apply this SE algorithm to answer candidates for list questions generated by Ephyra and other TREC QA systems to find additional instances of correct answers that were not in the original candidate set.

3 Proposed Approach

SEAL was originally designed to handle only relevant input seeds. When provided with a mixture of relevant and irrelevant answers from a QA system, the performance would suffer. In this section, we propose three modifications to SEAL to improve its ability to handle noisy input seeds.

3.1 Aggressive Fetcher

For each expansion, SEAL's fetcher concatenates all seeds and sends them as one query to the search engines. However, when the seeds are noisy, the documents fetched are constrained by the irrelevant seeds, which decreases the chance of finding good documents. To overcome this problem, we designed an *aggressive fetcher* (AF) that increases the chance of composing queries containing only relevant seeds. It sends a two-seed query for every possible pair of seeds to the search engines. If there are n input seeds, then the total number of queries sent would be $\binom{n}{2}$. For example, suppose SEAL is given a set of noisy seeds: *Boston*, *Seattle* and *Carnegie-Mellon* (assuming *Carnegie-Mellon* is irrelevant), then by using AF, one query will contain only relevant seeds (as shown in Table 1). The documents are then collected and sent to SEAL's extractor for learning wrappers.

	Queries	Quality
-AF	#1: Boston Seattle Carnegie-Mellon	Low
+AF	#1: Boston Seattle	High
	#2: Boston Carnegie-Mellon	Low
	#3: Seattle Carnegie-Mellon	Low

Table 1: Example queries and their quality given the seeds *Boston*, *Seattle* and *Carnegie-Mellon*, where *Carnegie-Mellon* is assumed to be irrelevant.

3.2 Lenient Extractor

SEAL's extractor requires the longest common contexts to bracket at least one instance of every seed per web page. However, when seeds are noisy, such common contexts usually do not exist or are too short to be useful. To solve this problem, we propose a *lenient extractor* (LE) which only requires the contexts to bracket at least one instance of *a minimum of two* seeds, instead of *every* seed. This increases the chance of finding longest common contexts that bracket only relevant seeds. For instance, suppose SEAL is given the seeds from the previous example (*Boston*, *Seattle* and *Carnegie-Mellon*) and the passage below. Then the extractor would learn the wrappers shown in Table 2.

"While attending a hearing in Boston City Hall, Alan, a professor at Boston University, met Tina, his former student at Seattle University, who is studying at Carnegie-Mellon University Art School and will be working in Seattle City Hall."

	Learned Wrappers
-LE	#1: `at [...] University`
+LE	#1: `at [...] University`
	#2: `in [...] City Hall`

Table 2: Wrappers learned by SEAL's extractor when given the passage in Section 3.2 and the seeds *Boston*, *Seattle* and *Carnegie-Mellon*.

As illustrated, with lenient extraction, SEAL is now able to learn the second wrapper because it brackets one instance of at least two seeds (*Boston* and *Seattle*). This can be very helpful if the list question is asking for city names rather than university names. The extractor then uses these wrappers to extract additional answer candidates, by searching for other strings that fit into the placeholders of the wrappers. Note that the example was simplified for ease of presentation. The wrappers are actually character-based (as opposed to word-based) and are likely to contain HTML tags when generated from real web pages.

3.3 Hinted Expander

Most QA systems use keywords from the question to guide the retrieval of relevant documents and the extraction of answer candidates. We believe these keywords are also important for SEAL to identify additional instances of correct answers. For example, if the seeds are *George Washington*, *John Adams*, and *Thomas Jefferson*, then without using any context from the question, SEAL would output a mixture of founding fathers and presidents of the U.S.A. To solve this problem, we devised a *hinted expansion* (HE) technique that utilizes the context given in the question to constrain SEAL's search space on the Web. This is achieved by appending keywords from the question to every query that is sent to the search engines. The rationale is that the retrieved documents will also match the keywords, which may increase the chance of finding those documents that contain our desired set of answers.

4 Experimental Design

We conducted experiments in two phases. In the first phase, we evaluated the SE approach by applying SEAL to answers generated by Ephyra. In the second phase, we evaluated the approach by applying SEAL to the output from QA systems that performed the best on the list questions in the TREC 15 evaluation. In both phases, the answers found by SEAL were retrieved from the Web instead of the AQUAINT newswire corpus used in the TREC evaluations. However, we rejected answers if they could only be found in the Web and not in the AQUAINT corpus to avoid an unfair advantage over the QA systems: TREC participants were allowed to extract candidates from the Web (or any other source), but they had to identify a supporting document in the AQUAINT corpus for each answer and thus could not return answers that were not covered by the corpus.

Preliminary experiments showed that we can obtain a good balance between the amount and quality of the documents fetched by using only rare question terms as hint words. In particular, we select the three question words that occur least frequently in a sample of the AQUAINT corpus as hints. The candidate answers were evaluated by using the answer keys, composed of regular expression patterns, obtained from the TREC website. We did not extend the patterns with additional correct answers found in our experiments. These answer keys were not officially used in the TREC evaluation; thus the baseline scores we computed for Ephyra and other QA systems in our experiments are slightly different from those officially reported.

4.1 Ephyra

We evaluated our SE approach on Ephyra using the list questions from TREC 13, 14, and 15 (55, 93, and 89 questions, respectively). For each question, the top four answer candidates from Ephyra were given as input seeds to SEAL. Initial experiments showed that by adding additional seeds, the effectiveness of our approach can be improved at the expense of a longer runtime.

We report both mean average precision (MAP) and F_1 scores. For the F_1 scores, we drop answer candidates with low confidence scores by applying a relative cut-off threshold: an answer candidate is dropped if the ratio of its confidence score and the score of the top answer is below a threshold. An optimal threshold for a question is a threshold that maximizes the F_1 score for that particular question.

For each TREC dataset, we conducted three experiments: (1) evaluation of answer candidates using MAP; (2) evaluation using average F_1 with an optimal threshold for each question; and (3) evaluation using average F_1 with thresholds trained by 5-fold cross validation. For each of those 5-fold validations, only one threshold was determined for all questions in the training folds.

	Ephyra	**Ephyra's Top 4 Ans.**	**SEAL**	**SEAL+LE**	**SEAL+LE +AF**	**SEAL+LE +AF+HE**
TREC 13	25.95%	21.39%	23.76%	31.43%	34.22%	35.26%
TREC 14	14.45%	8.71%	14.47%	17.04%	16.58%	18.82%
TREC 15	13.42%	9.02%	13.17%	16.87%	17.12%	18.95%

Table 3: Mean average precision of Ephyra, its top four answers, and various SEAL configurations, where LE is Lenient Extractor, AF is Aggressive Fetcher, and HE is Hinted Expander.

	Ephyra	**Ephyra's Top 4 Ans.**	**SEAL**	**SEAL+LE**	**SEAL+LE +AF**	**SEAL+LE +AF+HE**
TREC 13	35.74%	26.29%	30.53%	36.47%	40.08%	40.80%
TREC 14	22.83%	14.05%	20.62%	22.81%	22.66%	24.88%
TREC 15	22.42%	14.57%	19.88%	23.30%	24.04%	25.65%

Table 4: Average F_1 of Ephyra, its top four answers, and various SEAL configurations when using an optimal threshold for each question.

4.2 Top QA Systems

We evaluated two SE approaches, SEAL and Google Sets, on the five QA systems that performed the best on the list questions in TREC 15. For each question, the top four answer candidates[4] from those systems were given as input seeds to SEAL and Google Sets. Unlike the candidates found by Ephyra, these candidates were provided without confidence scores; hence, we assumed they all have a score of 1.0. In our experiments with SEAL, we first determined a single threshold that optimizes the average of the F_1 scores of the top five systems in both TREC 13 and 14. We then obtained evaluation results for the top systems in TREC 15 by using this trained threshold. When performing hinted expansion, the keywords (or hint words) for each question were extracted by Ephyra's question analysis component. In our experiments with Google Sets, we requested *Small Sets* of items and again measured the performance in terms of F_1 scores. We also tried requesting *Large Sets* but the results were worse.

5 Results and Discussion

In Tables 3 and 4, we present evaluation results for all answers from Ephyra, only the top four answers, and various configurations of SEAL using the top four answers as seeds. Table 3 shows the MAP for each dataset (TREC 13, 14, and 15), and Table 4 shows for each dataset the average F_1 score when using optimal per-question thresholds. The results indicate that SEAL achieves the best performance when configured with all three proposed extensions. In terms of MAP, the best-configured SEAL improves the quality of the input answers (relatively) by 65%, 116%, 110% for each dataset respectively, and improves Ephyra's overall performance by 36%, 30%, 41%. In terms of optimal F_1, SEAL improves the quality of the input answers by 55%, 77%, 76% and Ephyra's overall performance by 14%, 9%, 14% respectively. These results illustrate that a SE system is capable of improving a QA system's performance on list questions, if we know how to select good thresholds.

In practice, the thresholds are unknown and must be estimated from a training set. Table 5 shows evaluation results using 5-fold cross validation for each dataset (TREC 13, 14, and 15) independently, and the combination of all three datasets (All). For each validation, we determine the threshold that maximizes the F_1 score on the training folds, and we also determine the F_1 score on the test fold by applying the trained threshold. We repeat this validation for each of the five test folds and present the average threshold and F_1 score for each configuration and dataset. The F_1 scores give an estimate of the performance on unseen data and allow a fair com-

[4]Obtained from `http://trec.nist.gov/results`

	Ephyra		SEAL+LE+AF+HE		Hybrid	
	Avg. F_1	**Avg. Threshold**	**Avg. F_1**	**Avg. Threshold**	**Avg. F_1**	**Avg. Threshold**
TREC 13	25.55%	0.3808	30.71%	0.3257	29.04%	0.0796
TREC 14	15.78%	0.2636	15.60%	0.1889	17.13%	0.0108
TREC 15	15.19%	0.1192	15.64%	0.2581	16.47%	0.0123
All	18.03%	0.2883	19.15%	0.2606	19.59%	0.0164

Table 5: Average F_1 of Ephyra, the best-configured SEAL, and the hybrid system, along with thresholds trained by 5-fold cross validation.

TREC 15	**Baseline**	**Top 4 Ans.**	**Google Sets**		**SEAL+LE+AF+HE**		**Hybrid**	
QA Systems	**Avg. F_1**	**Avg. F_1**	**Avg. F_1**	ΔF_1	**Avg. F_1**	ΔF_1	**Avg. F_1**	ΔF_1
lccPA06	44.96%	32.67%	37.89%	-15.72%	40.00%	-11.04%	45.30%	0.76%
cuhkqaepisto	18.27%	17.02%	15.96%	-12.68%	19.75%	8.08%	19.13%	4.70%
NUSCHUAQA1	18.40%	14.99%	16.70%	-9.21%	18.74%	1.86%	18.06%	-1.81%
FDUQAT15A	19.71%	14.32%	18.79%	-4.63%	19.78%	0.38%	20.61%	4.57%
QACTIS06C	17.52%	15.22%	17.05%	-2.72%	18.45%	5.26%	18.38%	4.85%
Average	23.77%	18.84%	21.28%	-10.49%	23.34%	-1.81%	24.30%	2.20%

Table 6: Average F_1 of the QA systems, their top four answers, Google Sets, the best-configured SEAL, the hybrid system, and their relative improvements over the QA systems.

parison across systems. Here, we also introduce a hybrid system (Hybrid) that intersects the answers found by both systems by multiplying their probabilistic scores.

Tables 3, 4, and 5 show that the effectiveness of the SE approach depends on the quality of the initial answer candidates. The improvements are most apparent for the TREC 13 dataset, where Ephyra has a much higher performance compared to TREC 14 and 15. However, the best-configured SEAL did not improve the F_1 score on TREC 14, as reported in Table 5. We suspect that this is due to the comparatively low quality of Ephyra's top four answers for this dataset. The experiments also illustrate that by intersecting the answer candidates found by Ephyra and SEAL, we can eliminate poor answer candidates and partially compensate for the low precision of Ephyra on the harder TREC datasets. However, this comes at the expense of a lower recall, which slightly hurts the performance on the comparatively easier TREC 13 questions. We also evaluated Google Sets on top four answers from Ephyra for TREC 13-15 and obtained F_1 scores of 12%, 11%, and 9% respectively (compared to 29%, 17%, and 16% for our hybrid approach with trained thresholds).

Table 6 shows F_1 scores for the SE approach applied to the output from the five QA systems with the highest performance on the list questions in TREC 15. Again, Hybrid intersects the answers found by the QA system and SEAL by multiplying their confidence scores. Two thresholds were trained separately on the top five systems in both TREC 13 and 14; one for SEAL (0.2376) and another for Hybrid (0.2463). As shown, the performance of Google Sets is worse than SEAL and Hybrid, but better than the top four answers on average. We believe our SE system outperforms Google Sets because we have methods to handle noisy inputs (i.e. AF, LE) and a method for guiding the SE algorithm to search in the right space on the Web (i.e. HE).

The results show that both SEAL and Hybrid are capable of improving four out of the five systems. We observed that one reason why SEAL did not improve "lccPA06" was the incompleteness of the answer keys. Table 7 shows one of many examples where SEAL was penalized for finding additional correct answers. As illustrated, Hybrid improved all systems except "NUSCHUAQA1". The reason is that even though SEAL improved the baseline, their overlapping answer set is too small; thus hurting the recall of Hybrid substantially. Unfortunately, Question 154.6: Name titles of movies, other than "Superman" movies, that Christopher Reeve acted in.

lccPA06 (F_1: 75%)	SEAL+LE+AF+HE (F_1: 40%)
+Rear Window	+Rear Window
+The Remains of the Day	+The Remains of the Day
+Snakes and Ladders	-The Bostonians
-Superman	-Somewhere in Time
	-Village of the Damned
	-In the Gloaming

Table 7: Example of SEAL being penalized for finding correct answers (all are correct except the last one). Answers found in the answer keys are marked with "+". All four answers from "lccPA06" were used as seeds.

Question 170.6: What are the titles of songs written by John Prine?

NUSCHUAQA1 (F_1: 25%)	SEAL+LE+AF+HE (F_1: 44%)
+I Just Want to Dance With You	+I Just Want to Dance With You
-Titled In Spite of Ourselves	+Christmas in Prison
+Christmas in Prison	+Sam Stone
-Grammy - Winning	-Grandpa was a Carpenter
	-Sabu Visits the Twin Cities Alone
	+Angel from Montgomery

Table 8: Example demonstrating SEAL's ability to handle noisy input seeds. All four answers from "NUSCHUAQA1" were used as seeds. Again, SEAL is penalized for finding correct answers (all answers are correct).

for the top TREC 15 systems we only had access to the answers that were actually submitted by the participants, whereas for Ephyra we could utilize the entire list of generated answer candidates, including those that fell below the cutoff threshold for list questions. Nevertheless, the hybrid approach could improve the baseline by more than 2% on average in terms of F_1 score. Table 8 shows that the best-configured SEAL is capable of expanding only the relevant seeds even when given a set of noisy seeds. Neither Google Sets nor the original SE algorithm without the proposed extensions could expand these seeds with additional candidates.

On average, SEAL required about 5 seconds for querying the search engines, 10 seconds for crawling the Web, 20 seconds for extracting answer candidates from the web pages, and 5 seconds for ranking the candidates. Note that the SE system has not been optimized extensively. The runtime of the web page retrieval step and much of the search is due to network latency and can be reduced if the search is performed locally.

6 Conclusion and Future Work

We have shown that our SE approach is capable of improving the performance of QA systems on list questions by utilizing only their top four answer candidates as seeds. We have also illustrated a feasible and effective method for integrating a SE approach into any QA system. We would like to emphasize that for each of the experiments we conducted, all that the SE system received as input were the top four noisy answers from a QA system and three keywords from the TREC questions. We have shown that higher quality candidates support more effective set expansion. In the future, we will investigate how to utilize more answer candidates from the QA system and determine the minimal quality of those candidates required for SE approach to make an improvement.

We have also shown that, in terms of F_1 scores with trained thresholds, the hybrid method improves the Ephyra QA system on all datasets and also improves four out of the five systems that performed the best on the list questions in TREC 15. However, the final list of answers only comprises candidates found by both the QA system and the SE algorithm. In future experiments, we will investigate other methods of merging answer candidates, such as taking the union of answers from both systems. We expect further improvements from adding candidates that are found only by the QA system, but it is unclear how the confidence measures from the two systems can be combined effectively.

We would also like to emphasize that the SE approach is entirely language independent, and thus can be readily applied to answer candidates in other languages. In future experiments, we will investigate its performance on question answering tasks in languages such as Chinese and Japanese.

As pointed out previously, the performance of the SE approach highly depends on the accuracy of the seeds. However, QA systems are usually not optimized to provide few high-precision results, but treat precision and recall as equally important. This leaves room for further improvements, e.g. by applying stricter answer validation techniques to the seeds used for SE.

We also plan to analyze the effectiveness of our approach across different question types and evaluate it on more complex questions such as the rigid list questions in the new TAC QA evaluation, which ask for opinion holders and subjects.

Acknowledgements

This work was supported in part by the Google Research Awards program, IBM Open Collaboration Agreement #W0652159, and the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010.

References

H.T. Dang, J. Lin, and D. Kelly. 2006. Overview of the TREC 2006 question answering track. *Proceedings of the Fifteenth Text REtrieval Conference*.

H.T. Dang, D. Kelly, and J. Lin. 2007. Overview of the TREC 2007 question answering track. *Proceedings of the Sixteenth Text REtrieval Conference*.

J. Ko, L. Si, and E. Nyberg. 2007. A probabilistic framework for answer selection in question answering. *Proceedings of NAACL-HLT*.

N. Schlaefer, P. Gieselmann, and G. Sautter. 2006. The Ephyra QA system at TREC 2006. *Proceedings of the Fifteenth Text REtrieval Conference*.

N. Schlaefer, G. Sautter, J. Ko, J. Betteridge, M. Pathak, and E. Nyberg. 2007. Semantic extensions of the Ephyra QA system in TREC 2007. *To appear in: Proceedings of the Sixteenth Text REtrieval Conference*.

R.C. Wang and W.W. Cohen. 2007. Language-independent set expansion of named entities using the web. *Proceedings of IEEE International Conference on Data Mining*.

Acquiring Domain-Specific Dialog Information from Task-Oriented Human-Human Interaction through an Unsupervised Learning

Ananlada Chotimongkol
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
ananlada@cs.cmu.edu

Alexander I. Rudnicky
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
air@cs.cmu.edu

Abstract

We describe an approach for acquiring the domain-specific dialog knowledge required to configure a task-oriented dialog system that uses human-human interaction data. The key aspects of this problem are the design of a dialog information representation and a learning approach that supports capture of domain information from in-domain dialogs. To represent a dialog for a learning purpose, we based our representation, the *form-based dialog structure representation*, on an observable structure. We show that this representation is sufficient for modeling phenomena that occur regularly in several dissimilar task-oriented domains, including information-access and problem-solving. With the goal of ultimately reducing human annotation effort, we examine the use of unsupervised learning techniques in acquiring the components of the form-based representation (i.e. task, subtask, and concept). These techniques include statistical word clustering based on mutual information and Kullback-Liebler distance, TextTiling, HMM-based segmentation, and bisecting *K*-mean document clustering. With some modifications to make these algorithms more suitable for inferring the structure of a spoken dialog, the unsupervised learning algorithms show promise.

1 Introduction

In recent dialog management frameworks, such as *RavenClaw* (Bohus and Rudnicky, 2003) and *Collagen* (Rich et al., 2001), domain-dependent components of a dialog manager are clearly separated from domain-independent components. This separation allows rapid development of a dialog management module in a new task-oriented domain as dialog system developers can focus only on specifying domain-specific dialog information (e.g. the *Dialog Task Specification* in RavenClaw and *Task Models* in Collagen) while general dialog behaviors (e.g. turn-taking, confirmation mechanism, and generic help) are provided by the framework. For task-oriented domains, the domain-specific dialog information is equivalent to task-specific information. Examples of the task-specific information are steps in a task and domain keywords.

Specifying task-specific knowledge by hand is still a time consuming process (Feng et al., 2003). Furthermore, the hand-crafted knowledge may not reflect users' perceptions of a task (Yankelovich, 1997). To reduce the subjectivity of system developers, recorded conversations of humans performing a similar task as a target dialog system have been used to help the developers design the task specification. Nevertheless, analyzing a corpus of dialogs by hand requires a great deal of human effort (Bangalore et al., 2006). This paper investigates the feasibility of automating this dialog analysis process through a machine-learning approach. By inferring the task-specific dialog information automatically from human-human interaction data, the knowledge engineering effort could be reduced as the developers need to only revise learned information rather than analyzing a large amount of data.

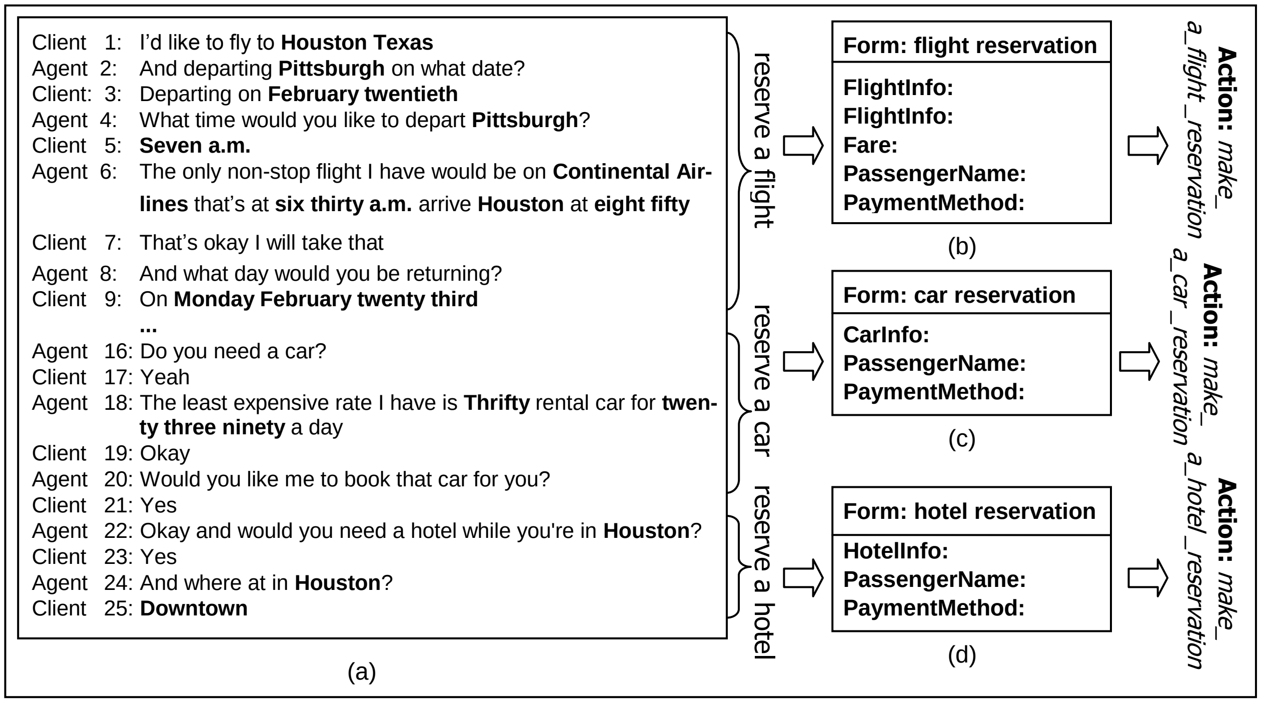
Acquiring the task-specific knowledge from a corpus of human-human dialogs is considered a knowledge acquisition process, where the target task structure has not yet been specified but will be explored from data before a dialog system is built. This is contrasted with a dialog structure recognition process (Alexandersson and Reithinger, 1997;

Bangalore et al., 2006; Hardy et al., 2004), where pre-specified dialog structure components are recognized as a dialog progresses.

We use an unsupervised learning approach in our knowledge acquisition process as it can freely explore the structure in the data without any influence from human supervision. Woszczyna and Waibel (1994) showed that when modeling a dialog state transition diagram from data an unsupervised approach outperformed a supervised one as it better reflects the characteristic of the data. It is also interesting to see how well a machine-learning approach can perform on the problem of task-specific knowledge acquisition when no assumption about the domain is made and no prior knowledge is used.

Examination of task-oriented human-human dialogs show that task-specific information can be observed in dialog transcription; therefore, it should be feasible to be infer it through an unsupervised learning approach. Figure 1 (a) shows a dialog in an air travel domain. This dialog is organized into three parts according to the three steps (i.e. reserve a flight, reserve a car, reserve a hotel) required to accomplish the task, creating a travel itinerary. Domain keywords (highlighted in bold) required to accomplish each step are clearly communicated.

To infer task-specific knowledge from data using an unsupervised learning approach, two problems need to be addressed: 1) choosing an appropriate dialog representation that captures observable task-specific knowledge in a dialog, and 2) developing an unsupervised learning approach that infers the task-specific knowledge modeled by this representation from in-domain human-human dialogs. The first problem is discussed in Section 3 where a *form-based dialog structure representation* is proposed. After describing the definition of each component in the form-based dialog structure representation, examples of how a domain expert models the task-specific information in a dialog with the form-based representation are given Section 3.1. Then the annotation experiment which was used to verify that the form-based representation can be understood and applied by other human annotators is discussed in Section 3.2. For the second problem, we modify existing unsupervised learning approaches to make them suitable for inferring the structure of a spoken dialog. Section 4 describes these modifications and their performances when inferring the components of the form-based dialog structure representation from interaction data.


Client 1: I'd like to fly to **Houston Texas**
Agent 2: And departing **Pittsburgh** on what date?
Client: 3: Departing on **February twentieth**
Agent 4: What time would you like to depart **Pittsburgh**?
Client 5: **Seven a.m.**
Agent 6: The only non-stop flight I have would be on **Continental Airlines** that's at **six thirty a.m.** arrive **Houston** at **eight fifty**
Client 7: That's okay I will take that
Agent 8: And what day would you be returning?
Client 9: On **Monday February twenty third**
...
Agent 16: Do you need a car?
Client 17: Yeah
Agent 18: The least expensive rate I have is **Thrifty** rental car for **twenty three ninety** a day
Client 19: Okay
Agent 20: Would you like me to book that car for you?
Client 21: Yes
Agent 22: Okay and would you need a hotel while you're in **Houston**?
Client 23: Yes
Agent 24: And where at in **Houston**?
Client 25: **Downtown**

Form: flight reservation
FlightInfo:
FlightInfo:
Fare:
PassengerName:
PaymentMethod:

Form: car reservation
CarInfo:
PassengerName:
PaymentMethod:

Form: hotel reservation
HotelInfo:
PassengerName:
PaymentMethod:

reserve a flight
reserve a car
reserve a hotel
Action: make_a_flight_reservation
Action: make_a_car_reservation
Action: make_a_hotel_reservation
(a)
(b)
(c)
(d)

Figure 1: An example of a dialog in the air travel domain and its corresponding form-based representation

2 Related Work

Automatic task-specific knowledge acquisition for configuring a dialog system is a relatively new research area. Supervised learning approaches were used to acquire a task model for a collaborative agent (Garland et al., 2001) and task-specific information for a customer care service (Feng et al., 2003). These supervised algorithms were trained on rich knowledge sources (examples described in a specific annotation language and a well-organized website respectively) annotated by domain experts. In contrast, the unsupervised conceptual clustering algorithm in DIA-MOLE (Möller, 1998) requires no additional human annotation to infer a set of domain-specific dialog acts from in-domain dialogs. The motivation behind the use of an unsupervised approach is similar to ours, to reduce human effort in creating a new dialog system.

3 Form-based Dialog Structure Representation

Many models have been proposed to account for the structure of a human-human conversation. Many such models focus on other aspects of a dialog such as coordinated activities, i.e. turn-taking and grounding, (Traum and Hinkelman, 1992) and regular patterns in the dialog (Carletta et al., 1997) rather than the domain-specific information communicated by participants. More complicated dialog representations (Grosz and Sidner, 1986; Litman and Allen, 1987) model several aspects of a dialog including domain-specific information. However, additional components in these models, such as beliefs and intentions, are difficult to observe directly from a conversation and, as for the current technology, may not be learnable through an unsupervised learning approach.

Since the task-specific information that we would like to model will be used for configuring a dialog system, we can view this information from a dialog system perspective. Our dialog representation is based on *form*, a data representation used in a form-based (or frame-based) dialog system. A form is a simple representation that captures necessary task-specific information communicated through dialog. This information is observable from dialog transcription (see below) and thus could be inferred through an unsupervised learning approach.

Typically, a form corresponds to a database query form while slots in the form represent search criteria. Nevertheless, a form can represent related pieces of information required to perform any domain action not just a database query action. With this more general definition of a form, a *form-based dialog structure representation* can be applied to various types of task-oriented domains where dialog participants have to gather pieces of information, analogous to search criteria, through dialog in order to perform domain actions that fulfill a dialog goal. Chotimongkol (2008) provided examples of these domains, for instance, meeting (Banerjee and Rudnicky, 2006) and flight simulation control (Gorman et al., 2003).

In the form-based dialog structure representation, task-specific information in each dialog is organized into a three-level structure of concept, subtask and task. A *concept* is a word or a group of words which captures a piece of information required to perform a domain action. A *subtask* is a subset of a dialog which contains sufficient concepts to execute a domain action that advances a dialog toward its goal. A *task* is a subset of a dialog (usually the entire dialog) which contains all the subtasks that belong to the same goal. A subtask can also be considered as a step in a task. In terms of representation, a task is represented by a set of forms, one for each of its subtasks. A concept is a slot in a form.

To model the structure of a dialog in a new domain with the form-based dialog structure representation, a list of tasks, subtasks, and concepts in that domain has to be specified. This list is considered a domain-specific tagset. The form-based dialog structure framework only provides the definitions of these components (i.e. task, subtask, and concept), which can be regarded as meta-tags and are domain-independent. A list of tasks, subtasks, and concepts can be identified manually as shown in Section 3.1 or automatically through a machine-learning approach as discussed in Section 4. Section 3.1 illustrates how a domain expert models the task-specific information in two task-oriented domains, air travel planning (information-accessing) and map reading (problem-solving), with the form-based representation. These examples also show that the form-based dialog structure representation is sufficient for modeling task-specific information in dissimilar domains.

Nonetheless, by focusing on observable task-specific information and describing this information using a simple model, the form-based dialog structure representation cannot model the information that is not clearly expressed in a dialog. Example of such information in an air travel domain is the pickup date of a car rental which may not be discussed in a dialog as it can be inferred from the arrival date of the corresponding flight. Furthermore, the form-based representation is not well suited for modeling a complex dialog that has a dynamic structure such as a tutoring dialog.

3.1 Dialog Structure Modeling Examples

Figure 1 illustrates how a dialog in the air travel domain (Eskenazi et al., 1999) can be represented with the form-based dialog structure representation. A dialog in this domain usually has a single goal, to create an air-travel itinerary which may include hotel and car reservations. Thus, the entire dialog corresponds to one task. The dialog in Figure 1 (a) contains three subtasks, one for each ***make_•_reservation*** action. The forms that represent these subtasks are shown in Figure 1 (b) – (d). Each form contains a set of concepts necessary for making the corresponding reservation. For a display purpose, the values of these slots are omitted.

A subtask can be further decomposed. For example, to reserve a round trip ticket, two database lookup actions, one for each leg, are required. A **reserve_flight** subtask in Figure 1 is decomposed into two **query_flight_info** subtasks. The corresponding forms of these subtasks are illustrated in Figure 2. Each **FlighInfo** concept in the flight reservation form is a result of a database lookup action that corresponds to each flight query form.

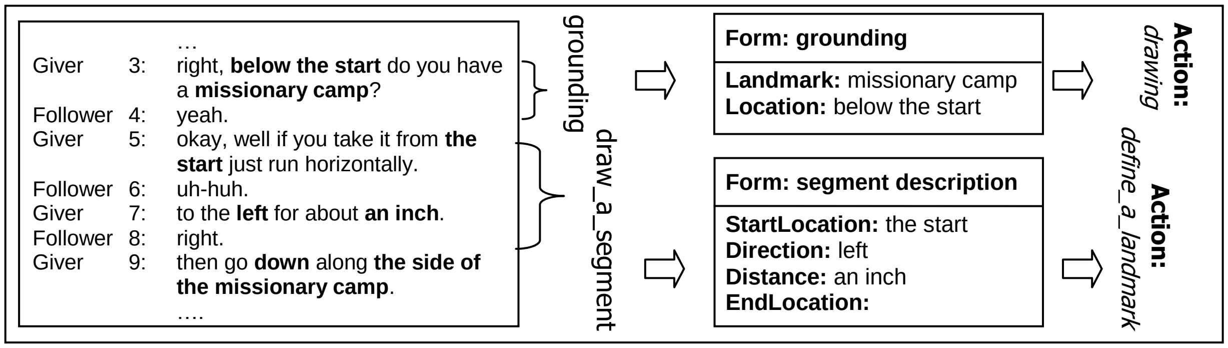
Form: flight query
DepartCity: Pittsburgh **ArriveCity:** Houston **ArriveState:** Texas **DepartDate:** February twentieth **DepartTime:** seven a.m.

Form: flight query
DepartCity: Houston **ArriveCity:** Pittsburgh **DepartDate:** Monday February twenty third **DepartTime:** five p.m.

Form: flight reservation
FlightInfo: **Airline:** Continental **DepartTime:** six thirty a.m. **ArriveCity:** Houston **ArriveTime:** eight fifty **FlightInfo:** **Airline:** Continental **DepartCity:** Houston **DepartTime:** six forty p.m. **ArriveCity:** Pittsburgh **ArriveTime:** ten twenty p.m **Fare:** four hundred dollars **Name:** **PaymentMethod:**

Figure 2: An example of subtask decomposition

Figure 3 show a dialog in the map reading domain (Anderson et al., 1991) and its corresponding form-based dialog structure representation. The goal of a dialog in this domain is to have a route follower draw a route on his/her map according to a description given by a route giver. Since drawing an entire route involves several drawing strokes, a **draw_a_route** task is divided into several **draw_a_segment** subtasks, one for each ***drawing*** action. This action required a set of concepts that describe a segment as shown in a segment description form. Since the landmarks on the giver's map can be different from those in the follower's map, the participants have to explicitly define the **Location** of a mismatched **Landmark** before using it in a segment description. In this case **grounding** becomes another subtask and can be represented by a form. This type of grounding is not necessary in the air travel domain.



		…
Giver	3:	right, **below the start** do you have a **missionary camp**?
Follower	4:	yeah.
Giver	5:	okay, well if you take it from **the start** just run horizontally.
Follower	6:	uh-huh.
Giver	7:	to the **left** for about **an inch**.
Follower	8:	right.
Giver	9:	then go **down** along **the side of the missionary camp**.
		….

Form: grounding
Landmark: missionary camp **Location:** below the start

Form: segment description
StartLocation: the start **Direction:** left **Distance:** an inch **EndLocation:**

Figure 3: An example of a dialog in the map reading domain and its corresponding form-based representation

3.2 Annotation Experiment

The goal of this annotation experiment is to verify that the form-based dialog structure framework can be understood by human annotators other than its developers, and that they can consistently apply the framework to model task-specific information in a dialog. In this experiment, each annotator had to design a form-based dialog structure representation for a given task-oriented domain by specifying a hierarchical structure of tasks, sub-tasks and concepts in that domain. Note that we are interested in the process of designing a domain-specific tagset from the definitions of task, subtask, and concept provided by the framework, not in the process of using an existing tagset to annotate data (see for example (Carletta et al., 1997)). The description of the framework is provided in annotation guidelines along with examples from the domains that were not used in the experiment.

The experimental procedure is as follows: the subjects first developed their own tagset according to the guidelines by analyzing a set of in-domain dialogs, and then annotate those dialogs with the tagset they had designed. To obtain enough annotated instances for each dialog structure component and to make the annotation simple, the dialog structure annotation part of the experiment was divided into two sub-parts: concept annotation and task/sub-task annotation. Two domains were used in the experiment, air travel planning and map reading. Four subjects were assigned to each domain. None had used the scheme previously. The average number of tags that each subject annotated is shown in the first row of Table 1.

Since some variations in tagset designs are acceptable as long as they conform to the guidelines, each subject's annotation is judged against the guidelines rather than one specific reference annotation. An annotation instance is marked as incorrect only when it does not conform to the guidelines. Each subject's annotation was evaluated by both a coding scheme expert and by other subjects. *Accuracy* is computed from the expert's judgment while *acceptability* is computed from peers' judgments. Acceptability scores shown in Table 1 were averaged from all other subjects in the same group. Please note that the result presented in this table should not be compared to the results from machine-learning approaches presented in Table 2 and Table 3 as the evaluation procedures and data sets are different.

Measure	**Air Travel**		**Map Reading**	
	C	**T**	**C**	**T**
Number of tags	178.8	50.5	347.8	60.8
Accuracy (%)	96.5	89.7	89.0	65.2
Acceptability (%)	95.6	81.1	94.9	84.5

Table 1: Accuracy and acceptability on concept annotation (C), and task/subtask annotation (T)

Both accuracy and acceptability are high for all annotation tasks except for the accuracy of task/subtask annotation in the map reading domain. Most of the errors come from the annotation of the **grounding** subtasks. Since its corresponding action is quite difficult to observe, subjects may not have a concrete definition of **grounding** and were more likely to produce errors. In addition, they were less critical when judging other subjects' annotations. Consistency in applying the form-based dialog structure representation shows that the representation is unambiguous and could potentially be identified through a machine-learning approach.

When comparing among components, concepts were annotated more consistently than tasks and subtasks in terms of both accuracy and acceptability. One possible reason is that, a concept is easier to observe as its unit is smaller than a task or a subtask. Moreover, dialog participants have to clearly communicate the concepts in order to execute a domain action. The subjects usually agreed on tasks and top-level subtasks, but did not quite agree on low-level subtasks. The low-level subtasks are correlated with the implementation of a dialog system; hence, the designs of these subtasks are more subjective and likely to be different.

4 Learning Approaches

This section describes machine-learning approaches for inferring the task-specific information modeled by the form-based dialog structure representation from human-human conversations. Specifically, the learning approach has to infer a list of tasks, sub-tasks and concepts in a given domain from in-domain dialogs similar to what a human does in Section 3. To make the problem tractable, components in the form-based representation are acquired separately. For most task-oriented dialogs that we encountered, each dialog corresponds to one task. Hence, the learning effort can be focused on identifying concept and subtask. Since we can only observe instances or values of these components in a dialog, we have to first identify these instances and then make a generalization for its type. For instant, to infer that there is a concept **City** in the air travel domain, a set of city names has to be identified and grouped together.

To identify a set of domain concepts from the transcription of in-domain dialogs, we follow the algorithm described in (Chotimongkol and Rudnicky, 2002). This algorithm utilizes an unsupervised clustering algorithm which clusters words based on context similarity, e.g. mutual information-based and Kullback-Liebler-based clustering, since the members of the same domain concept are usually used in similar contexts in a particular domain. Examples of the clusters obtained from the KL-based clustering algorithm are shown in Figure 4. These clusters represent **Hour**, **RentalCompany**, and **City** respectively. Underlined cluster members belong to other concepts. The clustering algorithm can identify all 12 members of **Hour** and about half of **RentalCompany**. In the third cluster, some airport names got merged with city names because they occur in quite similar context.

- ONE, TWO, THREE, NINE, SIX, FOUR, SEVEN, FIVE, EIGHT, TEN, TWELVE, ELEVEN
- HERTZ, BUDGET, THRIFTY
- MIDWAY, LAGUARDIA, GATWICK, PHILADELPHIA, DALLAS, DENVER, MONTEREY, BOSTON, CHICAGO, AUSTIN, NEWARK, PITTSBURGH, SEATTLE, OTTAWA, SYRACUSE, BALTIMORE, HOUSTON, MADRID, L.A., ATLANTA, DULLES, HONOLULU

Figure 4: Learned concepts in the air travel domain

The rest of this section describes an approach for identifying subtasks and their corresponding forms in a given domain. We decided to simplify the form-learning problem by first segmenting a dialog into form-filling episodes (which are equivalent to sub-tasks), then grouping the ones that correspond to the same form together so that we can determine a set of necessary slots in each form from the concepts present in its corresponding cluster. We further simplify the problem by concentrating on the domains that have only one top-level task (though in principle the approach can be extended to the domains that have multiple top-level tasks). Since we utilize well-known unsupervised algorithms, only the modifications which are applied to make these algorithms suitable for inferring the structure of a spoken dialog are discussed.

Two unsupervised discourse segmentation algorithms are investigated: TextTiling (Hearst, 1997) and Hidden Markov Modeling (Barzilay and Lee, 2004). These algorithms only recover the sequence of subtasks but not the hierarchical structure of subtasks similar to Bangalore et al.'s (2006) chunk-based model. Nevertheless, this simplification is sufficient when a subtask is embedded at the beginning or the end of the higher-level subtask which is the case for most embedded structures we have found. Both algorithms, while performing well with expository text, require modifications when applying to a fine-grained segmentation problem of spoken dialogs. In WSJ text, the average topic length is 428 words (Beeferman et al., 1999) while in the air travel domain the average subtask length is 84 words (10 utterances).

For TextTiling, the modifications include a distance weight and a data-driven stop word list. For the subtasks that are much shorter than the average length, distant words in the context window can be irrelevant. A *distance weight* demotes the importance of the context word that is far away from the considered boundary by giving it a lower weight. A manually prepared stop word list, containing common words, may not be suitable for every application domain. We propose a novel approach that determines a list of stop words directly from word distribution in each data set. TextTiling assumes that words that occur regularly throughout a dialog are not informative. However, the regularity of a particular word is determined from its distribution over the dialog rather than from its frequency. A high frequency word is useful if its instances occur only in a specific location. For example, the word "delta" which occurs many times in a **reserve_flight** subtask but does not occur in other subtasks is undoubtedly useful for determining subtask boundaries while the word "you" which can occur anywhere in a dialog is not useful. Specifically, a *regularity count* of word *w* is defined as the number of sliding context windows in the similarity score calculation of TextTiling that contain the word *w* in each dialog. A *data-driven stop word list* contains words that have a regularity count greater than a pre-defined threshold.

For HMM-based segmentation, we modified Barzilay and Lee's (2004) content models by using larger text spans when inducing the HMM states. HMM states are created automatically by clustering similar text spans together. When using an utterance as a text span, it may not contain enough information to indicate its relevant subtask as some utterances in a task-oriented dialog are very short and can occur in any subtask (e.g. acknowledgements and yes/no responses). Larger text spans, reference topics, were used in (Yamron et al., 1998). Nevertheless, this approach requires true segment boundaries. To eliminate the need of annotated data in our algorithm, HMM states are induced from predicted segments generated by TextTiling instead.

After segmenting all dialogs into sequences of subtasks, the bisecting *K*-means clustering algorithm (Steinbach et al., 2000) is used to group the segments that belong to the same type of subtask together as they represent the same form type. The clustering is done based on cosine similarity between segments. This unsupervised clustering algorithm is also used to infer a set of HMM states in the HMM-based segmentation described above.

Words are used as features for both segmentation and clustering algorithms. If a set of domain concepts has already been identified, we can use this information to enhance the features. When concept annotation is available, we can incorporate a concept label into a representation of a concept word. A *Label+Word* representation joins a word string and its label and can help disambiguate between similar words that belong to different concepts. For instance, "one" in "that one" is not the same token as "**[Hour]:**one". A *Label* representation, on the other hand, only represents a concept word by its label. This representation is based on the assumption that a list of concepts occurring in one subtask is distinguishable from a list of concepts occurring in other subtasks regardless of the values of the concepts; hence, a concept label is more informative than its value. This representation provides an abstraction over all different values of the same concept type. For example, **[Airline]:**northwest and **[Airline]:**delta are represented with the same token **[Airline]**. In all experiments, concept labels are provided by a domain expert as we assume that a set of domain concepts has already been identified.

4.1 Dialog Segmentation Results

To evaluate dialog segmentation performance, we compare predicted boundaries against subtask boundaries annotated by a domain expert. Subtask boundaries could occur only at utterance boundaries. Two metrics are used: P_k (Beeferman et al., 1999) and concept-based f-measure (C. F-1). P_k measures the probability of misclassifying two utterances that are *k* utterances apart as belonging to the same sub-task or different sub-tasks. *k* is set to half the average sub-task length. *C. F-1* is a modification of the standard f-measure (a harmonic mean of precision and recall) that gives credit to some near misses. Since the segmented dialogs will later be used to identify a set of forms and their associated slots, the segment that contains the same set of concepts as the reference segment is acceptable even if its boundaries are slightly different from the reference. For this reason, a near-miss counts as a match if there is no concept between the near-miss boundary and the reference boundary.

We evaluated the proposed dialog segmentation algorithms with 24 dialogs from the air travel domain and 20 dialogs from the map reading domain. The window size for TextTiling was set to 4 utterances. The cut-off threshold for choosing subtask boundaries was set to $\mu - \sigma/2$; where μ is the mean of the *depth scores* (Hearst, 1997), the relative change in word co-occurrence similarities on both sides of a candidate boundary, in each dialog and σ is their standard deviation. We found that a small window size and a low cut-off threshold are more suitable for identifying fine-grained segments as in the case of subtasks. However, we also found that TextTiling is quite robust as varying these two parameters doesn't severely degrade its performance (Chotimongkol, 2008). The threshold for selecting data-driven stop words was set to $\mu + 2*\sigma$; where μ is the mean of the regularity counts of all the words in a given dialog and σ is their standard deviation. The performance of TextTiling and HMM-based segmentation algorithm is shown in Table 2.

Augmented TextTiling, which uses a data-driven stop word list, distance weights, and the *Label+Word* representation, performed significantly better than the baseline in both domains. Each of these augmenting techniques can on their own improve segmentation performance but not significantly. Unsurprisingly, the proposed regularity counts discover stop words that are specific to spoken dialogs, but are absent from the hand-crafted list[1], e.g. “okay” and “yeah”.

Algorithm	Air Travel		Map Reading	
	P_k	C. F-1	P_k	C. F-1
TextTiling (baseline)	0.387	0.621	0.412	0.396
TextTiling (augmented)	0.371	0.712	0.384	0.464
HMM-based (utterance)	0.398	0.624	0.392	0.436
HMM-based (segment)	0.385	0.698	0.355	0.507
HMM-based (segment + Label representation)	0.386	0.706	0.250	0.686

Table 2: Dialog segmentation results

For HMM-based segmentation, the segmentation result obtained when modeling the HMM states from predicted subtasks generated by TextTiling (4th row) is better than the result obtained when modeling the HMM states from utterances (3rd row). Predicted segments provide more context to the clustering algorithm that induces the HMM states. As a result a more robust state representation is obtained. A more efficient clustering algorithm can also improve the performance of the HMM-based segmentation algorithm since it provides a state representation that better differentiates among dialog segments which belong to dissimilar subtasks. When the *Label* representation which yielded a better subtask clustering result (see Section 4.2) was used, HMM-based segmentation produced a better result (5th row) especially in the map reading domain. These numbers may appear modest compared to the numbers obtained when segmenting expository text. However, predicting the boundaries of fine-grained subtasks is more difficult even with a supervised learning approach (Arguello and Rosé, 2006). Our results are comparable to Arguello and Rosé’s (2006) results.

Between the two segmentation algorithms, the HMM-based algorithm performed slightly worse than TextTiling in the air travel domain but performed significantly better in the map reading domain. The HMM-based algorithm can identify more boundaries between fine-grained subtasks, which occur more often in the map reading domain. TextTiling, which relies on local lexical cohesion, is unlikely to find two significant drops in lexical similarity that are only a couple of utterances apart, and thus fails to detect boundaries of short segments. However, HMM-based segmentation misses more boundaries between two subtask occurrences of the same type, which occurs more often in the air travel domain, as they are usually represented by the same state.

[1] http://search.cpan.org/~creamyg/Lingua-StopWords-0.08/lib/Lingua/StopWords.pm.

4.2 Subtask Clustering Results

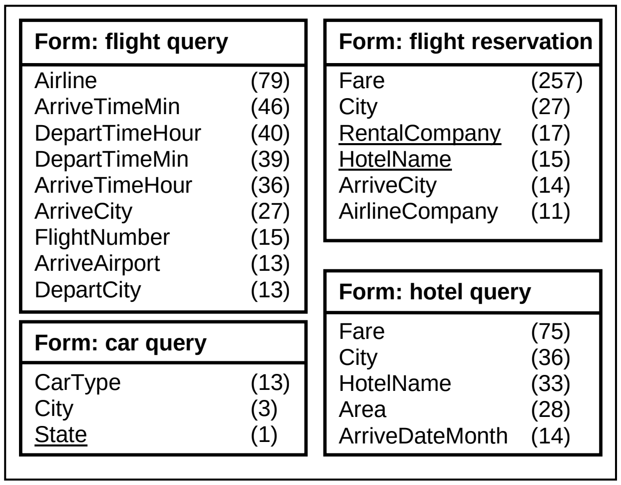
We evaluated the subtask clustering algorithm on the same data set used in the dialog segmentation evaluation. Table 3 presents the quality score (QS) for each clustering result. These QSs were obtained by comparing the output clusters against a set of reference subtasks. See (Chotimongkol and Rudnicky, 2002) for the definition of QS.

Feature Representation	Air Travel	Map Reading
Label+Word (oracle)	0.738	0.791
Label+Word	0.577	0.675
Label	0.601	0.823

Table 3: Subtask clustering results

When predicted segments were clustered, the quality of the output (2nd row) is not as good as when the reference segments were used (1st row) as inaccurate segment boundaries affected the performance of the clustering algorithm. However, the qualities of subtasks that occur frequently are not much different. In terms of feature representation, the clustering algorithm that uses the *Label* representation achieved better performance in both domains. When the sets of concepts in all of the subtasks are disjoint, the clustering algorithm that uses the *Label* representation can achieve a very good result as in the map reading domain. This result is even better than the result obtained when the reference segments were clustered by the algorithm that uses the *Label+Word* representation. These results demonstrate that an appropriate feature representation provides more useful information to the clustering algorithm than accurate segment boundaries. However, when the subtasks contain overlapping sets of concepts as in the air travel domain, the performance gain obtained from the *Label* representation is quite small.

Figure 5 shows four types of forms in the air travel domain that were acquired by the proposed form identification approach. The slot names are taken from concept labels. The number in parentheses is slot frequency in the corresponding cluster. The underlined slots are the ones that belong to other forms. Some slots in the car query form are missing as some instances of its corresponding subtask get merged into other clusters.



Form: flight query	
Airline	(79)
ArriveTimeMin	(46)
DepartTimeHour	(40)
DepartTimeMin	(39)
ArriveTimeHour	(36)
ArriveCity	(27)
FlightNumber	(15)
ArriveAirport	(13)
DepartCity	(13)

Form: car query	
CarType	(13)
City	(3)
State	(1)

Form: flight reservation	
Fare	(257)
City	(27)
RentalCompany	(17)
HotelName	(15)
ArriveCity	(14)
AirlineCompany	(11)

Form: hotel query	
Fare	(75)
City	(36)
HotelName	(33)
Area	(28)
ArriveDateMonth	(14)

Figure 5: Examples of forms obtained by the proposed unsupervised learning approach

4.3 Discussions on Learning Approaches

The results presented in the previous sections show that existing unsupervised learning algorithms are able to identify components of the form-based dialog structure representation.. However, some modifications are required to make these algorithms more suitable for inferring the structure of a spoken dialog. The advantages of different learning algorithms can be combined to improve performance. For example, TextTiling and HMM-based segmentation are good at detecting different types of boundaries; therefore, combining the predictions made by both algorithms could improve segmentation performance. Additional features such as prosodic features could also be useful.

Subsequent steps in the learning process are subjected to propagation errors. However, the proposed learning algorithms, which are based on generalization of recurring patterns, are able to learn from inaccurate information given that the number of errors is moderate, so that there are enough correct examples to learn from. Given redundant information in dialog corpora, a domain knowledge acquisition process does not require high learning accuracy and an unsupervised learning approach is reasonable. The overall quality of the learning result is acceptable. The proposed unsupervised learning approach can infer much useful task-specific dialog information needed for automatically configuring a task-oriented dialog system from data.

5 Conclusion and Future Directions

To represent a dialog for a learning purpose, we based our representation, the form-based dialog structure representation, on observable information. Components of the form-based representation can be acquired with acceptable accuracy from observable structures in dialogs without requiring human supervision. We show that this dialog representation can capture task-specific information in dissimilar domains. Additionally, it can be understood and applied by annotators other than the developers.

Our investigation shows that it is feasible to automatically acquire the domain-specific dialog information necessary for configuring a task-oriented dialog system from a corpus of in-domain dialogs. This corpus-based approach could potentially reduce human effort in dialog system development. A limitation of this approach is that it can discover only information present in the data. For instance, the corpus-based approach cannot identify city names absent in the corpus while a human developer would know to include these. Revision may be required to make learned information more accurate and complete before deployment; we expect that this effort would be less than the one required for manual analysis. A detailed evaluation of correction effort would be desirable.

In this paper, task-specific knowledge was acquired from in-domain dialogs without using any prior knowledge about the domain. In practice, existing knowledge sources about the world and the domain, such as WordNet, could be used to improve learning. Some human supervision can be valuable particularly in the form of semi-supervised learning and active learning. In particular a process that integrates human input at appropriate times (for example seeding or correction) is likely to be part of a successful approach.

Acknowledgments

This research was supported by DARPA grant NB CH-D-03-0010. The content of the information in this publication does not necessarily reflect the position or the policy of the US Government, and no official endorsement should be inferred.

References

J. Alexandersson and N. Reithinger. 1997. Learning Dialogue Structures From A Corpus. In *Proceedings of EuroSpeech-97*. Rhodes, Greece.

A. H. Anderson, M. Bader, E. G. Bard, E. Boyle, G. Doherty, S. Garrod, S. Isard, J. Kowtko, J. McAllister, J. Miller, C. Sotillo, H. Thompson, and R. Weinert. 1991. The HCRC Map Task Corpus. *Language and Speech,* 34(4):351-366.

J. Arguello and C. P. Rosé. 2006. Topic Segmentation of Dialogue. In *Proceedings of Workshop on Analyzing Conversations in Text and Speech*.

S. Banerjee and A. I. Rudnicky. 2006. You Are What You Say: Using Meeting Participants' Speech to Detect their Roles and Expertise. In *the NAACL-HLT 2006 workshop on Analyzing Conversations in Text and Speech*. New York, NY.

S. Bangalore, G. D. Fabbrizio, and A. Stent. 2006. Learning the Structure of Task-Driven Human-Human Dialogs. In *Proceedings of COLING/ACL 2006*. Sydney, Australia.

R. Barzilay and L. Lee. 2004. Catching the Drift: Probabilistic Content Models, with Applications to Generation and Summarization. In *Proceedings of HLT-NAACL 2004*. Boston, MA.

D. Beeferman, A. Berger, and J. Lafferty. 1999. Statistical Models for Text Segmentation. *Machine Learning,* 34(1-3):177-210.

D. Bohus and A. I. Rudnicky. 2003. RavenClaw: Dialog Management Using Hierarchical Task Decomposition and an Expectation Agenda. In *Proceedings of Eurospeech2003*. Geneva, Switzerland.

J. Carletta, S. Isard, G. Doherty-Sneddon, A. Isard, J. C. Kowtko, and A. H. Anderson. 1997. The reliability of a dialogue structure coding scheme. *Computational Linguistics*, 23(1):13-31.

A. Chotimongkol. 2008. *Learning the Structure of Task-Oriented Conversations from the Corpus of In-Domain Dialogs,* Ph.D. Thesis CMU-LTI-08-001. Pittsburgh, Carnegie Mellon University.

A. Chotimongkol and A. Rudnicky. 2002. Automatic Concept Identification in Goal-Oriented Conversations. In *Proceedings of ICSLP 2002*. Denver, CO.

M. Eskenazi, A. Rudnicky, K. Gregory, P. Constantinides, R. Brennan, C. Bennett, and J. Allen. 1999. Data Collection and Processing in the Carnegie Mellon Communicator. In *Proceedings of Eurospeech 1999*. Budapest, Hungary.

J. Feng, S. Bangalore, and M. Rahim. 2003. WebTalk: Mining Websites for Automatically Building Dialog Systems. In *Proceedings of ASRU '03*. St. Thomas, U.S. Virgin Islands.

A. Garland, N. Lesh, and C. Sidner. 2001. Learning Task Models for Collaborative Discourse. In *Proceedings of Workshop on Adaptation in Dialogue Systems, NAACL '01*. Pittsburgh, PA.

J. C. Gorman, N. J. Cooke, P. W. Foltz, P. A. Kiekel, and M. J. Martin. 2003. Evaluation of Latent Semantic Analysis-based measures of team communications content. In *Proceedings of the Human Factors and Ergonomic Society 47th Annual Meeting, (HFES 2003)*.

B. J. Grosz and C. L. Sidner. 1986. Attention, Intentions, and the Structure of Discourse. *Computational Linguistics,* 12(3):175-204.

H. Hardy, A. Biermann, R. B. Inouye, A. Mckenzie, T. Strzalkowski, C. Ursu, N. Webb, and M. Wu. 2004. Data-Driven Strategies for an Automated Dialogue System. In *Proceedings of ACL '04*. Barcelona, Spain.

M. A. Hearst. 1997. TextTiling: Segmenting Text into Multi-paragraph Subtopic Passages. *Computational Linguistics*, 23(1):33-64.

D. Litman and J. Allen. 1987. A Plan Recognition Model for Subdialogues in Conversations. *Cognitive Science,* 11(2):163-200.

J.-U. Möller. 1998. Using Unsupervised Learning for Engineering of Spoken Dialogues. In *Proceedings of AAAI 1998 Spring Symposium on Applying Machine Learning to Discourse Processing*.

C. Rich, C. L. Sidner, and N. Lesh. 2001. Collagen: applying collaborative discourse theory to human-computer interaction. *AI Magazine,* 22(4):15-25.

M. Steinbach, G. Karypis, and V. Kumar. 2000. A Comparison of Document Clustering Techniques. In *Proceedings of KDD Workshop on Text Mining*.

D. R. Traum and E. A. Hinkelman. 1992. Conversation Acts in Task-Oriented Spoken Dialogue. *Computational Intelligence*, 8(3):575--599.

M. Woszczyna and A. Waibel. 1994. Inferring linguistic structure in spoken language. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*.

J. P. Yamron, I. Carp, L. Gillick, S. Lowe, and P. v. Mulbregt. 1998. A Hidden Markov Model Approach to Text Segmentation and Event Tracking. In *Proceedings of ICASSP '98*. Seattle, WA.

N. Yankelovich. 1997. Using Natural Dialogs as the Basis for Speech Interface Design. In Susann Luperfoy (Ed.), *Automated Spoken Dialog Systems*. Cambridge, MA: MIT Press.

Relative Rank Statistics for Dialog Analysis

Juan M. Huerta
IBM T.J. Watson Research Center
1101 Kitchawan Road
Yorktown Heights, NY 10598
huerta@us.ibm.com

Abstract

We introduce the relative rank differential statistic which is a non-parametric approach to document and dialog analysis based on word frequency rank-statistics. We also present a simple method to establish semantic saliency in dialog, documents, and dialog segments using these word frequency rank statistics. Applications of our technique include the dynamic tracking of topic and semantic evolution in a dialog, topic detection, automatic generation of document tags, and new story or event detection in conversational speech and text. Our approach benefits from the robustness, simplicity and efficiency of non-parametric and rank based approaches and consistently outperformed term-frequency and TF-IDF cosine distance approaches in several experiments conducted.

1 Background

Existing research in dialog analysis has focused on several specific problems including dialog act detection (e.g., Byron and Heeman 1998), segmentation and chunking (e.g., Hearst 1993), topic detection (e.g., Zimmerman et al 2005), distillation and summarization (e.g., Mishne et al 2005) etc. The breath of this research reflects the increasing importance that dialog analysis has for multiple domains and applications. While historically, dialog analysis research has initially leveraged the corresponding techniques originally intended for textual document analysis, techniques tailored specifically for dialog processing eventually should be able to address the sparseness, noise, and time considerations intrinsic to dialog and conversations.

The approach proposed in this paper focuses on the relative change of rank ordering of words occurring in a conversation according to their frequencies. Our approach emphasizes relatively *improbable* terms by focusing on terms that are relatively unlikely to appear frequently and thus weighting their change in rank more once they are observed. Our technique achieves this in a non-parametric fashion without explicitly computing probabilities, without the assumption of an underlying distribution, and without the computation of likelihoods.

In general, non-parametric approaches to data analysis are well known and present several attractive characteristics (as a general reference see Hollander and Wolfe 1999). Non-parametric approaches require few assumptions about the data analyzed and can present computational advantages over parametric approaches especially when the underlying distributions of the data are not normal. In specific, our approach uses rank order statistics of word-feature frequencies to compute a relative rank-differential statistic.

This paper is organized as follows: in Section 2 we introduce and describe our basic approach (the relative rank differential RRD function and its sorted list). In Section 3 we address the temporal nature of dialogs and describe considerations to dynamically update the RRD statistics in an on-line fashion especially for the case of shifting temporal windows of analysis. In Section 4 we relate the RRD approach to relevant existing and previous dialog and text analysis approaches. In Section 5 we illustrate the usefulness of our metric by analyzing a set of conversations in various ways using the RRD. Specifically, in that section we will empirically demonstrate its robustness to noise and data sparseness compared to the popular term frequency and TF-IDF cosine distance approaches in a dialog classification task. And finally, in Section 6, we present some concluding remarks and future directions

2 The Relative Rank Differential

Let $d^u = \{d_1{}^u, d_2{}^u, d_3{}^u ...\}$ denote the ranked dictionary of a language (i.e., the ordered list of words sorted in decreasing order of frequency). The superscript *u* denotes that this ranked list is based on the *universal* language. Specifically, the word $d_i{}^u$ is the i^{th} entry in d^u if its frequency of occurrence in the language denoted by $f(d_i{}^u)$ is larger than $f(d_j{}^u)$ for every j where $i < j$ (for notational simplicity we assume that no two words share the same frequency). In the case where want to relax this assumption we simply allow $i < j$ when $f(d_i^u) = f(d_j{}^u)$ as long as d_i^u precedes $d_j{}^u$ lexicographically, or under any other desired precedence criteria. For d^u we assume that $f(d_i^u) > 0$ for every entry (i.e., each word has been observed at least once in the language).

Similarly, let now $d^S = \{d_1{}^S, d_2{}^S, d_3{}^S ...\}$ denote the corresponding ranked dictionary for a dialog, or dialog segment, S (ordered, as in the case of the language dictionary, in decreasing order of frequency)[1]. The superscript S denotes that this ranked list is based on the dialog *S*. The word $d_i{}^S$ is the i^{th} entry in d^s if its frequency of occurrence in the conversation segment *S* denoted by $f(d_i{}^S)$ is larger than $f(d_j{}^S)$ for every j where $i < j$. In this case we allow $f(d_i{}^S) \geq 0$ for every i so that the cardinality of d^u is the same as d^s.

Let $r_d(w)$ denote the rank of word w in the ranked dictionary d so that, for example, $r_{d^u}(d_i^u) = i$.

Based now on a dialog segment and a universal language, any given word w will be associated with a rank in d^u (the universal ranked dictionary) and a rank in d^s, the dialog segment ranked dictionary.

Let us define now for every word the relative rank differential (RRD) function or statistic[2] given by:

$$c_{d^s, d^u}(w) = \frac{\left| r_{d^u}(w) - r_{d^s}(w) \right|}{\left(r_{d^u}(w) \right)^{\alpha}}$$

The relative rank-differential is the ratio of the absolute difference (or change) in rank between the word's original position in the universal dictionary and the segment s. The exponent α in the denominator allows us to emphasize or deemphasize changes in terms according to their position or rank in the language (or universal) dictionary. Typically we will want to increase the denominator's value (i.e., deemphasize) for terms that have very low frequency (and their rank value in the universal dictionary is large) so that only relatively big changes in rank will result in substantial values of this function.

When alpha is zero, the RRD focuses on every word identically as we consider only the absolute change in rank. For alpha equal to 1.0 the relative change in rank gets scaled down linearly according to its rank, while for alphas larger than 1.0 the numerator will scale down or reduce to a larger extent the value of relative rank differential for words that have large rank value.

Based on each word's relative rank differential we can compute the ranked list of words sorted in decreasing order by their corresponding value of relative rank differential. Let this sorted list of words be denoted by $R(d^u, d^S) = \{w_1, w_2, ...\}$. So that $c(w_i)$ is larger than $c(w_j)$[3] for every j where $i < j$.

We now provide some intuition on the ranked RRD lists and the RRD function. The ranked dictionary of a language contains information

[1] We only consider at this point the case in which both speakers' parts in a dialog interaction are considered jointly (i.e., single channel), however, our method can be easily extended to separate conversation channels. Also, for simplicity we consider at this point only words (or phrases) as features.

[2] For brevity, we refer to the Relative Rank Differential of a word given two utterances as a *statistic*. It is not, strictly speaking, a metric or a distance, but rather a function.

[3] For simplicity, c is written without subscripts when these are apparent from the context.

about the frequency of all words in a language (i.e., across the universe of conversations) while the segment counterpart pertains a single conversation or segment thereof. The relative rank differential tells us how different a word is ranked in a conversation segment from the universal language, but this difference is normalized by the universal rank of the word. Intuitively, and especially when alpha equals 1.0, the RRD denotes some sort of percent change in rank. This also means that this function is less sensitive to small changes in frequency in the case of frequent words and to small changes in rank in case of infrequent words. Finally, the sorted list $R(d^u, d^S)$ contains in order of importance the most relatively salient terms of a dialog segment, as measured by relative changes or differences in rank.

3 Collecting Rank Statistics

We now discuss how to extend the metrics described in the previous section to consider finite-time sliding windows of analysis, that is, we describe how to update rank statistics, specifically the ranked lists and relative rank differential information for every feature in an on-line fashion. This is useful when tracking the evolution of single dialogs, when focusing the analysis to span shorter regions, as well as to supporting dynamic real-time analytics of large number of dialogs.

To approach this, we decompose the *word events* (words as they occur in time) into arriving and departing events. An arriving event at time *t* is a word that is covered by the analysis window at its specific time as the finite length window slides in time, and a departing word at time *t* is a feature that stops falling within the window of analysis. For simplicity, and without loss of generality, we now assume that we are performing the analysis in real time and that the sliding window of analysis spans from current time *t* back to *(t-T)*, where *T* is the length of the window of analysis.

An arriving word at current time *t* falls into our current window of analysis and thus needs to be processed. To account for these events efficiently, we need a new structure: the temporal event *FIFO* list (i.e., a queue where events get registered) that keeps track of events as they arrive in time. As an event (word w_t) arrives it is registered and processed as follows:

1. Find the corresponding identifier of w_t in the universal ranked dictionary and add it as d_i^u at the end of the temporal event list together with its time stamp.
2. The corresponding entry in d^s, the ranked segment dictionary, is located through an index list that maps $d_i^u \rightarrow d_k^s$ and the segment frequency associated is incremented $f(d_k^s) = f(d_k^s) + 1$
3. Verify if the rank of the feature needs to be updated in the segment rank list. In other words evaluate whether $f(d_{k-1}^s) > f(d_k^s)$ still holds true after the update. If this is not true then shift feature up in the rank list (to a higher rank) and shift down the predecessor feature in the rank list. In this single shift-up-down operation, update the index list and the value of *k*.
4. For every feature shifted down in 3 down re-compute the relative rank differential RRD function and verify if its position needs to be modified in $R(d^u, d^S)$ (a second index list is needed to compute this efficiently).
5. Repeat step 3 iteratively until feature is not able to push up any further in the ranked list.

The process for dealing with departing events is quite similar to the arriving process just described. Of course, as the analysis window slides in time, it is necessary to keep track of the temporal event FIFO list to make sure that the events at the top are removed as soon as they fall out of the analysis window. The process is then:

1. The departing event is identified and its corresponding identifier in the universal ranked dictionary d_i^u is removed from the top of the temporal event list.
2. Its location in d^s the ranked segment dictionary is located through the index list. The corresponding segment frequency associated is decreased as follows: $f(d_k^s) = f(d_k^s) - 1$.
3. Verify if the rank of the feature needs to be updated in the segment rank list. In other

words evaluate if $f(d^s_{k+1}) < f(d^s_k)$ still holds true after the update. If not shift feature down in rank (to a lower rank, denoting less frequent occurrence) and shift the successor feature up in the rank list. In this single shift up-down operation, update the index list and the value of k.

4. For every feature shifted up in step 3 recompute the relative rank differential and verify if its location needs to be modified in $R(d^u, d^S)$
5. Repeat step 3 until the feature is not able to shift down any further in the ranked list.

The procedures just described are efficiently implementable as they simply identify entries in rank lists through index lists, update values by incrementing and decrementing variables, and performed some localized and limited re-sorting. Additionally, simple operations like adding data at the end and removing data at the beginning of the FIFO list are needed making it altogether computationally inexpensive.

4 Related Techniques

Our work relates to several existing techniques as follows. Many techniques of text and dialog analysis utilize a word frequency vector based approach (e.g., Chu-Carroll et al 1999) in which lexical features counts (term frequencies) are used to populate the vector. Sometimes the term frequency is normalized by document size and weighted by the inverse document frequency (TF-IDF). The TF-IDF and TF metrics are the base of other approaches like discriminative classification (Kuo and Lee 2003; and Li and Huerta 2004), Text Tilling or topic chains (Hearst 1993; Zechner 2001), and latent semantic indexing (Landauer et al 1998). Ultimately, these types of approaches are the foundation of complex classification and document understanding systems which use these features together with possibly more sophisticated classification algorithms (e.g., D'Avanzo et al 2007).

When using TF and TF-IDF approaches, it is important to notice that by normalizing the term frequency by the document length, TF-based approaches are effectively equivalent to estimation of a multinomial distribution. The variance of the estimate will be larger as the number of observations decreases. Recently, approaches that explicitly establish this parametric assumption and perform parameter inference have been presented in (Blei et al 2003). This work is an example of the potential complexity associated when performing parameter inference.

The area of adaptation of frequency parameters for ASR, specifically the work of (Church 2000), is relevant to our work in the sense that both approaches emphasize the importance of and present a method to update the lexical or semantic feature statistics on-line.

In the area of non-parametric processing of dialog and text, the work of (Huffaker et al 2006), is very close to the work in this paper as it deals with non-parametric statistics of the word frequencies (rank of occurrences) and uses the Spearman's Correlation Coefficient. Our work differs from this approach in two ways: first, the Relative Rank Differential tells us about the relative change in rank (while SCC focuses in the absolute change) and secondly, from the ranked RDD list, we can identify the saliency of each term (as opposed to simply computing the overall similarity between two passages).

5 Experiments

In order to illustrate the application of the RRD statistic, we conducted two sets of experiments based on conversations recorded in a large customer contact center for an American car manufacturer. In the first group of experiments we took a corpus of 258 hand transcribed dialogs and conducted classification experiments using the basic RRD statistic as feature. We compared its performance against term frequency and TF-IDF based cosine distance approaches. The second set of experiments is based on ASR transcribed speech and for this we used a second corpus consisting of a set of 44 conversations spanning over 3 hours of conversational speech.

In the first set of experiments we intend to illustrate two things: first the usefulness of RRD as a feature in terms of representational accuracy and second, its robustness to noise and data sparseness compared to other popular features. In the second set of experiments we illustrate the versatility and potential of our technique to be applied in dialog-oriented analysis.

5.1 RRD for Dialog matching

For this set of experiments we used a corpus of 258 hand transcribed conversations. Each dialog was treated like a single document. Using the set of dialogs we constructed different query vectors and affected these queries using various noise conditions, and then we utilized these vectors to perform a simple document query classification experiment. We measured the cosine distance between the noisy query vector and the document vector of each document in this corpus. A noisy query is constructing by adding zero mean additive gaussian noise to the query vector with amplitude proportional to the value of a parameter *N* and with floor value of zero to avoid negative valued features. We allow, in these experiments, for counts to have non-integer values; as the dialog becomes larger, the Gaussian assumption holds true due to the Central Limit Theorem, independently of the actual underlying distribution of the noise source. This distortion is intended to mimic the variation between two similar dialogs (or utterances) that are essentially similar, except for a additive zero mean random changes. A good statistic should be able to show robustness to these types of distortions. A correct match is counted when the closest match for each query is the generating document.

	N=0.0	N=.05	N=0.1	N=0.2	N=0.4
TF-cosine	99.6	98.0	84.9	60.0	32.5
TF-IDF cosine	99.6	99.6	97.3	88.0	67.4
RRD-dot	99.6	99.6	97.6	91.8	70.9

Table 1. Query match accuracy for 3 features under several query noise conditions.

Table 1 shows the percent correct matches for the TF, TF-IDF and Relative Rank Differential features, under various levels of query noise. As we can see, in clean conditions the accuracy of the 3 features is quite high but as the noise conditions increase the accuracy of the 3 techniques decreases substantially. However, the TF feature is much more sensitive to noise than the other two techniques. We can see that our technique is better than both TF and TF-IDF in noisy conditions.

We also conducted experiments to test the comparative robustness or the RRD feature to query data sparseness. To measure this, we evaluated the accuracy in query-document match when using a random *subset* of the document as query. Figure 1 show the results of this experiment using the RRD feature, the Term Frequency, and the TF-IDF feature vectors. We can see that with as little as 5% of the document size as query, the RRD achieves close to 90% accuracy while the TF-IDF feature needs up to 20% to achieve the same performance, and the TF counts only need close to 70%.

These results empirically demonstrate that RRD statistics are more robust to noise and to term coverage sparseness than TF and TF-IDF.

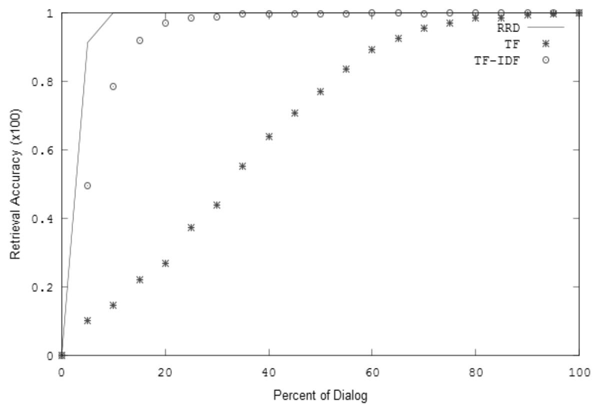


Figure1. Query match accuracy for 3 feature types under various query data sparseness conditions

5.2 ASR Based experiments

For the experiments of this section we used 44 dialogs. Manual transcriptions for these 44 conversations were obtained in order to evaluate the speech recognition accuracy. While we could have used the manual transcripts to perform the analysis, the results reported here are based on the recognition output. The reason for using ASR transcripts as opposed to human transcription is that we wanted to evaluate how useful our approach would be in a real ASR based solution dealing with large amounts of noisy data at this level of ASR error.

Each dialog was recorded in two separate channels (one for the agent and one for the customer) and automatically transcribed separately using a large vocabulary two-stage automatic speech recognizer system. In the first stage, a speaker independent recognition pass is performed after which the resulting hypothesis is used to compensate and adapt feature and models. Using the adapted feature and models the second stage recognition is performed. After recognition, the single best hypothesis with time stamps for the agent and customer are weaved back together.

The overall Word Error Rate is about 24% and varies significantly between the set of agents and the set of customers (the set of agents being more accurate).

The universal dictionary we used consists exclusively of the words occurring in the corpus which total 2046 unique words. Call length ranged from just less than 1 minute to more than 20 minutes with most of the calls lasting between 2 and 3 minutes. The corpus consists of close to 30k tokens and does not distinguish between agent channel and customer channel. A universal dictionary of ranked words is built from the set of dialogs and each dialog is treated as a segment.

Dialog Tagging and Topic Saliency

In this analysis we look at complete dialogs. A useful application of the methods we describe in this work is to identify and separate calls that are interesting from non-interesting calls[4], furthermore, one could also be interested in singling out which specific terms make this dialog salient. An application of this approach is the automatic generation of tags (e.g., social-network style of document tagging). In our approach, we will identify calls whose top entries in their sorted relative rank differential lists are above a certain threshold and deem these calls as semantically salient.

We now describe in detail how an interesting call can be distinguished from a non-interesting call using the relative rank differential statistic.

Figure 2 below shows the ranked dictionary $d^S = \{d_1^S, d_2^S, d_3^S ...\}$ (i.e., the universal rank id's as a function of their observed ranks) and Figure 3 shows the plot of the sorted relative rank differential list $R(d^u, d^S)$ for when the segment corresponds to an interesting call (as defined above).

The chosen call, specifically shows as topic AIRBAG deployment in the context of a car accident.

Specifically, Figure 2 shows the corresponding rank in the universal ranked dictionary versus the rank in the dialog or segment. We can see that the right-most part of the plot is largely monotonic, meaning that most entries of lesser frequency occur in the same ranked order both in the universal and the specific dialog (including zero times for the segment), while a subset across the whole range of the universal dictionary were substantially relocated up in the rank (i.e., occurred more frequently in the dialog than in the language). If the plot was a single straight line each word would have the same rank both in the language and in the dialog.

We argue that while the terms of interest lie in that subset of interest in the graph (the terms whose rank increased substantially), not all of those words are equally interesting or important and rather than simply looking at absolute changes in rank we focus on the relative-rank differential RRD metric. Thus, Figure 3 shows the sorted values of the relative rank differential list (with $\alpha = 1.3$). The top entries and their rank in the universal dictionary (in parentheses) are: AIRBAGS (253), AS (55), FRONT (321), DEPLOY (369), SIDE (279), ACCIDENT (687). As we can see, the top entries are distributed across a broad range of ranks in the universal dictionary and relate to the topic of the conversation, which from the top ranked entries are evidently the deployment of front and side airbags during an accident, and thus, for this call were able to identify its semantic saliency from the corpus of conversations.

Other interesting or salient calls also showed a similar this profile in the RRD curve.

The question now is what the behavior of our approach for uninteresting calls is. We repeated the procedure above for a call which we deemed semantically un-interesting (i.e., dealing with a common topic like call transfer and other routine procedures). Figure 4 shows the sorted relative rank differential values and, especially when compared with Figure 2, we see a large monotonic component on the higher ranked terms and not so marked discontinuities in the low and mid-range part of the curve.

We computed the relative rank differential RRD metric for each feature similarly as with the interesting call, and ranked the words based on these values. The distribution of the ranked values is shown in Figure 5. The resulting words with top values are CLEAR (1113), INFORMATION (122) BUYING (1941), and CLEARLY (1910). From these words we cannot really tell what is the specific topic of the conversation is as easily as with

[4] For the purpose of this work, we simply define as an *interesting call* a call that deals with an infrequent or rare topic which influences the distribution of keywords and key-phrases. Examples of calls in our domain meeting this criterion are calls dealing with accidents and airbags.

the interesting call. More importantly, we can now compare Figures 3 and 5 and see that the highest relative rank differential value of the top entry in Figure 3 (larger than 10) is significantly larger than the largest relative rank differential value in Figure 5 (just above 7) reflecting the fact that the relative rank differential metric could be a useful parameter in evaluating semantic saliency of a segment using a static threshold. As an interesting point, conceivably the highly ranked features based on RRD could reflect language utilization idiosyncrasies.

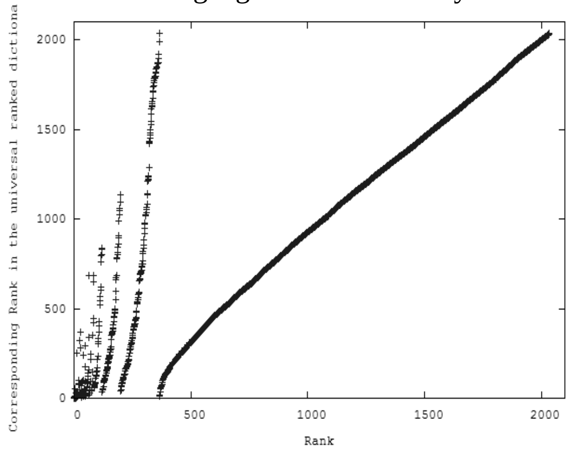


Figure 2. Ranked dictionary entry vs Universal Rank for a salient call

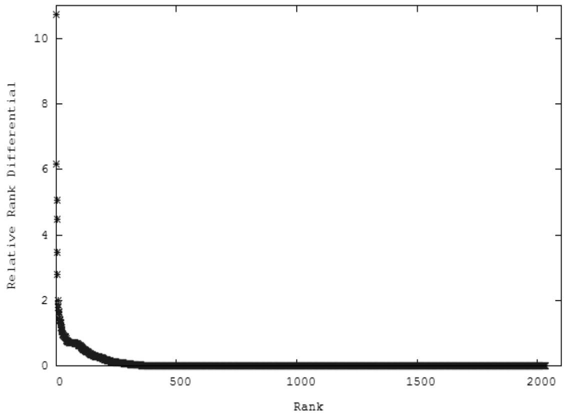


Figure 3. Sorted relative rank differential values of $R(d^u, d^S)$ for a semantically salient call.

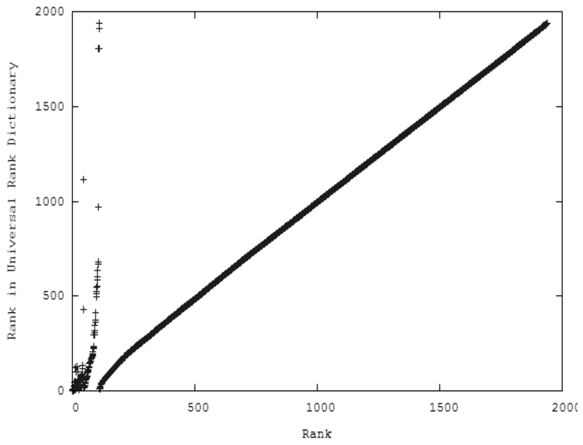


Figure 4. Ranked dictionary entry vs Universal Rank for a non-salient call

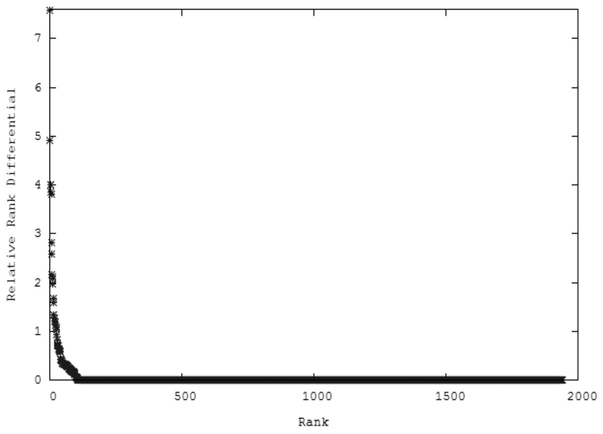


Figure 5. Sorted relative rank differential values of $R(d^u, d^S)$ for a non-interesting (semantically non-salient) call.

6 Conclusions

In this paper we presented a novel non parametric rank-statistics based method for the semantic analysis of conversations and dialogs. Our method is implementable in segment-based or dialog-based modalities, as well as in batch form and in on-line or dynamic form. Applications of our method include topic detection, event tracking, story/topic monitoring, new-event detection, summarization, information filtering, etc. Because our approach is based on non-parametric statistics it has favorable intrinsic benefits, like making no assumptions about the underlying data, which makes it suitable for the use of both lexical semantic features as well as classifier-based semantic features. Furthermore, our approach could, in the future, benefit from classical non-parametric approaches like block-treatment analysis etc.

We demonstrated that our approach is as effective in query classification as TF and TF-IDF in low noise and no noise (i.e., distortion) conditions, and consistently better than those techniques in noisy conditions. We also found RRD to be more robust to query data sparseness than TF and TF-IDF. These results provide a motivation to combine our statistic with other techniques like topic chains, textilling, latent semantic indexing, and discriminant classification approaches; specifically RRD could replace TF and TF-IDF based features.

Future work could focus on applying ranking statistics to techniques for mining and tracking temporal and time-changing parameters in conjunction with techniques like (Agrawal and Srikant 1995; Pratt 2001; Last et al 2001).

Another area of possible future work is the detection and separation of multiple underlying trends in dialogs. Our approach is also suited for the analysis of large streams of real time conversations, and this is a very important area of focus as presently more and more conversational data gets generated through channels like chat, mobile telephony, VoIP etc.

References

Agrawal R. and Srikant R. 1995. *Mining Sequential Patterns*. In Proc. of the 11th Int'l Conference on Data Engineering, Taipei, Taiwan.

Berger, A. L., Pietra, V. J., and Pietra, S. A. 1996. *A maximum entropy approach to natural language processing*. Comp. Linguist. 22, 1

Blei D., Ng A., and Jordan M. 2003. *Latent Dirichlet allocation*. J. of Machine Learning Research

Byron, D. K. and Heeman, P. A. 1998. *Discourse Marker Use in Task-Oriented Spoken Dialog*. TR664, University of Rochester.

Chu-Carroll, J, and Carpenter R. 1999. *Vector-Based Natural Language Call Routing*. Journal of Computational Linguistics, 25(30), pp. 361-388

Church, K. 2000. *Empirical estimates of adaptation: The chance of two Noriega 's is closer to p/2 than p2*. In Coling

D'Avanzo E., Elia A., Kuflik T., Vietri S. 2007. *University of Salerno, LAKE System at DUC 2007,* Proc. Document Understanding Conference

Hearst, M. 1993. *TextTiling: A Quantitative Approach to Discourse Segmentation*, Technical Report UCB:S2K-93-24, Berkeley, CA

Hollander & Wolfe 1999. *Nonparametric Statistical Methods*, Second Edition, John Wiley and Sons

Huffaker, D., Jorgensen, J., Iacobelli, F., Tepper, P., & Cassell, J. 2006. *Computational Measures for Language Similarity across Time in Online Communities*. Workshop on ACTS at HLT-NAACL, New York City, NY.

Klinkenberg R. and Renz I. 1998. *Adaptive information filtering: Learning in the presence of concept drifts*. In Learning for Text Categorization, Menlo Park

Kuo H.-K.J. and Lee C. H. *Discriminative training of natural language call routers*, IEEE Transactions on Speech and Audio Processing, Volume 11, Issue 1, Jan 2003 Page(s): 24 - 35.

Landauer T., Foltz P. W., and Laham D. *Introduction to Latent Semantic Analysis* Discourse Processes 25, 1998.

Last M., Klein Y., and Kandel A., *Knowledge Discovery in Time Series Databases* IEEE Trans. on Systems, Man, and Cybernetics 31B(2001).

Li X. and Huerta J.M., *Discriminative Training of Compound word based Multinomial Classifiers for Speech Routing* Proc. ICSLP 2004

Mishne, G., Carmel, D., Hoory, R., Roytman, A., and Soffer, A. 2005. *Automatic analysis of call-center conversations*. In *Proc. of the 14th ACM international Conference on information and Knowledge.*

Pratt K. B. *Locating patterns in discrete time series*. Master's thesis, Computer Science and Engineering, University of South Florida, 2001.

Stanley K. O. *Learning concept drift with a committee of decision trees*. Comp. Science Dept., University of Texas-Austin. TR AI-03-302, 2003.

Zechner K. *Automatic Summarization of Spoken Dialogues in Unrestricted Domains*. PhD thesis, LTI, CMU, 2001

Zimmermann M., Liu Y., E. Shriberg, and A. Stolcke, *Toward Joint Segmentation and Classification of Dialog Acts in Multiparty Meetings* MLMI workshop, 2005

Learning to Predict Code-Switching Points

Thamar Solorio and **Yang Liu**
Human Language Technology Research Institute
The University of Texas at Dallas
Richardson, TX 75080, USA
tsolorio,yangl@hlt.utdallas.edu

Abstract

Predicting possible code-switching points can help develop more accurate methods for automatically processing mixed-language text, such as multilingual language models for speech recognition systems and syntactic analyzers. We present in this paper exploratory results on learning to predict potential code-switching points in Spanish-English. We trained different learning algorithms using a transcription of code-switched discourse. To evaluate the performance of the classifiers, we used two different criteria: 1) measuring precision, recall, and F-measure of the predictions against the reference in the transcription, and 2) rating the naturalness of artificially generated code-switched sentences. Average scores for the code-switched sentences generated by our machine learning approach were close to the scores of those generated by humans.

1 Introduction

Multilingual speakers often switch back and forth between languages when speaking or writing, mostly in informal settings. The mixing of languages involves very elaborated patterns and forms and we usually use the term Code-Switching (CS) to encompass all of them (Lipski, 1978). Before the Internet era, CS was mainly used in its spoken form. But with so many different informal interaction settings, such as chats, forums, blogs, and web sites like Myspace and Facebook, CS is being used more and more in written form. For English and Spanish, CS has taken a step further. It has become a hallmark of the chicano culture as it is evident by the growing number of chicano writers publishing work in Spanish-English CS.

We have not completely discovered the process of human language acquisition, especially dual language acquisition. Findings in linguistics, sociolinguistics, and psycholinguistics show that the production of code-switched discourse requires a very sophisticated knowledge of the languages being mixed. Some theories suggest bilingual speakers might have a third grammar for processing this type of discourse. The general agreement regarding CS is that switches do not take place at random and instead it is possible to identify rules that bilingual speakers adhere to.

Understanding the CS process can lead to accurate methods for the automatic processing of bilingual discourse, and corpus-driven studies about CS can also inform linguistic theories. In this paper we present exploratory work on learning to predict CS points using a machine learning approach. Such an approach can be used to reduce perplexity of language models for bilingual discourse. We believe that CS behavior can be learned by a classifier and the results presented in this paper support our belief.

One of the difficult aspects of trying to predict CS points is how to evaluate the performance of the learner since switching is intrinsically motivated and there are no forced switches (Sankoff, 1998b). Therefore, standard classification measures for this task such as precision, recall, F-measure, or accuracy, are not the best approach for measuring the effectiveness of a CS predictor. To complement the evaluation of our approach, we designed a task involving human judgements on the naturalness of automatically generated code-switched sentences. Both evaluations yielded encouraging results.

The next section discusses theories explaining the CS production process. Then in Section 3 we present our framework for learning to predict CS points. Section 4 discusses the empirical evaluation of the classifiers compared to the human reference. In Section 5 we present results of human evaluations on automatically generated code-switched sentences. Section 6 describes previous work related to the processing of code-switched text. Finally, we conclude in Section 7 with a summary of our findings and directions for future work.

2 Bilingual Discourse

The combination of languages can be considered to be a continuous spectrum where on each end of the spectrum we have one of the standard languages and no blending. As one moves closer to the middle of the spectrum the amount and complexity of the blending pattern increases. The blending pattern most widely known, and studied, is code-switching, which refers to the mixing of words from two languages, but the words themselves do not suffer any syntactic or phonological alterations. The CS points can lie at sentence boundaries, but very often we will also observe CS inside sentences. According to (Sankoff, 1998b; Poplack, 1980; Lipski, 1978) when CS is used inside a sentence, it can only happen at syntactic boundaries shared by both languages, and the resulting monolingual fragments will conform to the grammar of the corresponding language. In this CS theory the relationship between both languages is symmetric –lexical items from one language can be replaced by the corresponding items in the second language and vice versa. Another prevalent linguistic theory argues the contrary: there is an asymmetric relation where the changes can occur only in one direction, which reflects the existence of a Matrix Language (ML), the dominant language, and an Embedded Language (EL), or subordinate language (Joshi, 1982). The Matrix Language Frame model, proposed and extended by Scotton-Myers, supports this asymmetric relation theory. This formalism prescribes that content morphemes can come from the ML or the EL, whereas late system morphemes, the elements that indicate grammatical relations, can only be provided by the ML (Myers-Scotton, 1997).

Until an empirical evaluation is carried out on large representative samples of discourse involving a large number of different speakers, and different language-pairs, the production of CS discourse will not be explained satisfactorily. The goal of this work is to move closer to a better understanding of CS by learning from corpora to predict possible CS points.

3 Learning When To Code-Switch

3.1 The English-Spanish Code-Switched Data Set

We recorded a conversation among three English-Spanish bilingual speakers that code-switch regularly when speaking to each other. The conversation lasts for about 40 minutes (∼8k words, 922 sentences). It was manually transcribed and annotated with Part-of-Speech (POS) tags. A total of 239 switches were identified manually. English is the predominant language used, with a total of 576 monolingual sentences. We refer to this transcription as the Spanglish data set. We are currently in the process of collecting new transcriptions of this conversation in order to measure inter annotator agreement.

3.2 Approach

Machine learning algorithms have proven to be surprisingly good at language processing tasks, including optical character recognition, text classification, named entity extraction, and many more. The premise of our paper is that machine learning algorithms can also be successful at learning how to code-switch as well as humans. At the very least we want to provide encouraging evidence that this is possible. To the best of our knowledge, there is no previous work related to the problem of automatically predicting CS points. Our machine learning framework then is inspired by existing theories of CS and existing work on part-of-speech tagging code-switched text (Solorio and Liu, 2008).

In our approach, each word boundary is a potential point for switching – an instance of the learning task. It should be noted that we can only rely on the history of words preceding potential CS points in or-

Feature id	Description
1	Word
2	Language id
3	Gold-standard POS tag
4	BIO chunk
5	English Tree Tagger POS
6	English Tree Tagger prob
7	English Tree Tagger lemma
8	Spanish Tree Tagger POS
9	Spanish Tree Tagger prob
10	Spanish Tree Tagger lemma

Table 1: Features explored in learning to predict CS points.

der to extract meaningful features. Otherwise, if we look also into the future, we could just do language identification to extract the CS points. However, our goal is to provide methods that can be used in real time applications, where we do not have access to observations beyond the point of interest. Another restriction we imposed on the method is related to the size of the context used. A sentence can be code-switched in different ways, with all different versions adhering to the CS "grammar". The number of permissible CS sentences grows almost exponentially with the length of the sentence[1]. By limiting the length of the context to at most two words we are trying to avoid some sort of over fitting by having the model making assumptions over the interaction of the two languages that will be too weak, or speaker-dependent.

Previous studies have identified several socio-pragmatic functions of code-switching. The most common include direct quotation, emphasis, clarification, parenthetical comments, tags, and trigger switches. Other characteristics relevant to CS behavior are the topic being discussed, the speakers involved, the setting where the conversation is taking place, and the level of familiarity between the speakers. Having encoded information regarding the CS function and the aforementioned relevant factors might help in predicting upcoming CS points. However, annotating this information in the transcription can be time consuming and very often this information is not readily available. Therefore, at the expense of making this task even more difficult, we decided against trying to include this type of information and include only lexical and syntactic features, to evaluate a practical and cost effective method for this task. Table 1 shows the list of features. All of these features are associated with word w_n, the word immediately preceding boundary n. Feature 1 is the word form[2]. Feature 2 is language identification. If the production of CS discourse adheres to the matrix language frame model, then knowledge of the language can potentially be a good source of information. Feature 3 is the gold-standard POS tag. We also include as a feature the position of the word relative to the phrase constituent using a Beginning-Inside-Outside (BIO) scheme. For instance, the word at the beginning of the verb phrase will be labeled as B, the following words inside this verb phrase will be tagged as I, and words that were not identified as part of a phrase constituent were labeled as O. This chunking information was extracted using the English and Spanish versions of FreeLing[3]. We did not measure accuracy on the chunking information. Features 5 to 9 were generated by tagging the Spanglish conversation using the Spanish and the English versions of the Tree Tagger (Schmid, 1994). Attributes 5 to 7 are extracted from the English version, which include the POS tag, the confidence, and the lemma for that word. Similarly, features 8 to 10 were taken from the Spanish monolingual tree tagger. Features from the monolingual taggers will have some noisy labels when tagging fragments of the other language. However, considering that our feature set is small we want to explore if adding these features, which include the lemmas and probability estimates, can contribute to the learning task.

We also explored using a larger context. In this case, we extract the same features shown in Table 1 for the two words preceding the word boundary, resulting in 20 attributes representing each instance.

Evaluation for this task is not straightforward. Within a sentence, there are several CS points that will result in a natural sounding code-switched sentence, but none of these CS points are mandatory.

[1]Almost exponentially because not all sentences will be considered grammatical.

[2]Strictly speaking these should be called tokens, not words since punctuation marks are considered as well.

[3]http://garraf.epsevg.upc.es/freeling/

CS has a lot to do with the speaker's preferences, the topic being discussed, and the background of the participants involved. Using the standard approach for measuring performance of classifiers can be misleading, especially if the reference data set is small and/or has only a small number of speakers. It is unrealistic to just consider F-measure, or accuracy, as truthfully reflecting how well the learners generalize to the task. Therefore, we evaluated the classifier's performance using two different criteria, which are discussed in the next sections.

4 Evaluation 1: Using the Reference Data Set

This is the standard evaluation of machine learning classifiers. We randomly divided the data into sentences and grouped them into 10 subsets to perform a cross-validation. Tables 2 and 3 show results for Naive Bayes (NB) and Value Feature Interval (VFI) (Demiroz and Guvenir, 1997). Using WEKA (Witten and Frank, 1999), we experimented with different subsets of the attributes and two context windows: using only the preceding word and using the previous two words. The results presented here are overall averages of 10-fold cross validation. We also report standard deviations. It should be noted that the Spanglish data set is highly imbalanced, around 96% of the instances belong to the negative class. Therefore, our comparisons are based on Precision, Recall, and F-measure, leaving accuracy aside, since a weak classifier predicting that all instances belong to the negative class will reach an accuracy of 96%.

The performance measures shown on Tables 2 and 3 show that NB outperforms VFI in most of the configurations tested. In particular, NB yields the best results when using a 1 word context with no lexical forms nor lemmas as attributes (see Table 2 row 3). This is a fortunate finding –for most practical problems there will always be words in the test set that have not been observed in the training set. For our small Spanglish data set that will certainly be the case. In contrast, VFI achieves higher F-measures when using a context of two words and all the features are used.

Analyzing the predictions of the learners we noted that the NB classifier is heavily biased by the language attribute, close to 80% of the positive predictions made by NB are after seeing a word in Spanish. This preference seems to support the assumption of the asymmetry between the two languages and the existence of an ML[4]. This however is not the case for VFI, only a little over 50% of the positive predictions belong to this scenario. Another interesting finding is the learner's tendency to predict a code-switch after observing words like "Yeah", "anyway", "no", and "shower". The first two seem to fit the pattern of idiomatic expressions. According to Montes-Alcalá this type of CS includes linguistic routines and fillers that are difficult to translate accurately (Montes-Alcalá, 2007), which might be the case of "anyway", and unconscious changes, which can explain the case of "Yeah". The case of "shower" and "no" are more difficult to explain, they might be overfitting patterns from the learners. We also found out that VFI learned to predict that a CS will take place right after seeing the sequence of words *le dije* (I said). This sequence of words is frequently used when the speaker is about to quote his/herself, and this quotation is one of the well-documented CS functions (Montes-Alcalá, 2007).

A greedy search approach for attribute selection using WEKA showed that out of the 20 attributes (when using a two word context), the subset with the highest predictive value included the language identification for word w_{n-1} and w_{n-2}, the confidence threshold from the English tagger for word w_{n-2}, the lemma from the Spanish Tree tagger for w_{n-1}, and the lexical form of the word w_{n-1}. We expected the chunk information to be useful and this does not seem to be the case. Another unexpected outcome is that higher F-measures are reached by adding features generated by the monolingual Tree taggers. Even though these features are noisy, they still carry useful information.

We only show results from NB and VFI. Initial experiments with a subset of the data showed that these algorithms were the most promising for this task. They both yielded higher F-measures, even when compared against Support Vector Machines (SVMs), C4.5, and neural networks. On this experiment all the discriminative classifiers reached a classification accuracy close to 96%, but an F-

[4]We remind the reader that in this paper ML stands for Matrix Language.

Features Used											**Naive Bayes**		
					English Tree tagger			Spanish Tree tagger					
C	Word Form	Lang id	POS tag	BIO chunk	POS	Prob	Lem	POS	Prob	Lem	**P**	**R**	**F_1**
1		X	X	X							0.09(0.01)	0.01(0.00)	0.02(0.00)
1	X	X	X	X							0.23(0.01)	0.32(0.02)	0.27(0.02)
1*		X	X	X	X	X		X	X		0.19(0.00)	0.53(0.00)	0.28(0.00)
1	X	X	X	X	X	X	X	X	X	X	0.18(0.00)	0.59(0.00)	0.27(0.00)
2		X	X	X							0.13(0.00)	0.35(0.00)	0.19(0.00)
2	X	X	X	X							0.16(0.00)	0.46(0.00)	0.23(0.00)
2		X	X	X	X	X		X	X		0.14(0.00)	0.55(0.01)	0.23(0.00)
2	X	X	X	X	X	X	X	X	X	X	0.16(0.00)	0.59(0.01)	0.25(0.00)

Table 2: Prediction results of CS points with NB using different features. Column C indicates the size of the context used, 1 indicates a 1 word context, and 2 indicates two words preceding the word boundary. Columns P, R, and F_1, show precision, recall, and F-measure, respectively. Numbers in parenthesis show standard deviations. The row marked with a '*' shows the configuration used for the generation of CS sentences presented in Section 5.

measure on the positive class of around 0%. NB and VFI estimate predictions for each class separately, which makes them robust to imbalanced data sets. In addition, generative models are known to be better for smaller data sets since they reach their higher asymptotic error much faster than discriminative models (Ng and Jordan, 2002). This might explain why Naive Bayes outperformed strong classifiers such as SVMs by a large margin.

The overall prediction performance is not very high. However, we should remark that for this particular task expecting a high F-measure is unrealistic. Consider for example, a case where the learners predict a CS point where the speaker decided not to switch, this does not imply that particular point is not a good CS point. And similarly, if the classifier missed an existing CS point in the reference data set the resulting sentence might still be grammatical and natural sounding. This motivated the use of an alternative evaluation, which we discuss below.

5 Evaluation 2: Using Human Evaluators

The goal of this evaluation is to explore how humans perceive our automatically generated CS sentences, and in particular, how do they compare to the original sentences and to the randomly generated ones. We selected 30 spontaneous and naturally occurring CS sentences from different sources. Some of them were selected from the Spanglish Times Magazine[5], some others from blogs found in (Montes-Alcalá, 2007). Other sentences were taken from a paper discussing CS on e-mails (Montes-Alcalá, 2005). All of the sentences are true occurrences of written CS, from speakers different from the ones in the Spanglish data set. The sentences were translated to standard English and Spanish and were manually aligned. We will use this parallel set of sentences to predict CS points with our models. Based on the model predictions we will generate code-switched sentences by combining monolingual fragments.

It should be noted that the Spanglish data set is a transcription of spoken CS. In contrast, this new evaluation set contains only written CS. Recent studies suggest written CS will adhere to the rules of spoken CS (Montes-Alcalá, 2005), but there is still some controversy on this issue. From our perspective, both samples come from informal conversational interactions. It is expected that both will have similar patterns and therefore will provide a good source for our evaluation.

5.1 Automatically Generated Code-Switching Sentences

In this subsection we describe how to generate code-switched sentences randomly and with the learned models described in the previous sections. For the

[5]http://www.spanglishtimes.com/

Features Used											Voting Feature Intervals		
					English Tree tagger			Spanish Tree tagger					
C	Word Form	Lang id	POS tag	BIO chunk	POS	Prob	Lem	POS	Prob	Lem	**P**	**R**	**F**$_1$
1		X	X	X							0.12(0.00)	0.68(0.00)	0.21(0.00)
1	X	X	X	X							0.12(0.00)	0.65(0.01)	0.20(0.00)
1*		X	X	X	X	X		X	X		0.12(0.00)	0.72(0.01)	0.21(0.00)
1	X	X	X	X	X	X	X	X	X	X	0.13(0.00)	0.65(0.00)	0.22(0.00)
2		X	X	X							0.13(0.00)	0.60(0.00)	0.21(0.00)
2	X	X	X	X							0.15(0.00)	0.52(0.01)	0.23(0.00)
2		X	X	X	X	X		X	X		0.13(0.00)	0.68(0.00)	0.22(0.00)
2	X	X	X	X	X	X	X	X	X	X	0.15(0.00)	0.51(0.00)	0.24(0.00)

Table 3: Prediction results of CS points with VFI using different features. The notation on this table is the same as in Table 2

classifier-based approach, we POS tagged each parallel set of sentences, with the monolingual English and Spanish Tree Taggers, and we extracted the same set of features described shown in Table 1. We decided to train the models with a context size of one word, even though both learners reached higher F-measures when using a two-word context. This decision was based on the observation that having a two-word context will pose restrictions on possible CS points, since we would not be able to switch unless we have inserted into the sentence at least two tokens from the same language.

We trained the NB and VFI models with the Spanglish data set (using features 2–6, 8, and 9, see Table 1) and generated CS predictions for each parallel file. A code-switched sentence is generated by adding the first token of the sentence in language 1 (L1), and continue adding more tokens from L1 until a CS point is found. When a CS prediction is found, the following tokens are selected from the second language (L2), and we continue adding tokens from L2 until the classifier has predicted a change. Different versions of the sentences are generated by changing the definition of L1 and L2.

For the randomly generated CS sentences, switching decisions are made randomly with a probability proportional to the positive predictions made by the classifiers (in this case NB). That is, for the Spanish sentences switch points are predicted randomly with a 30% chance of switching while for English switch points are predicted with a 10% chance.

Generator	Average Score
Human	3.64
NB	3.33
Random	2.68
VFI	2.50

Table 4: Average score of 18 judges over the set of 28 code-switched sentences rated.

In total we generated 180 CS sentences: 30 sentences per generator scheme (we have three generators: NB, VFI, and random), and two versions from each generator corresponding to the two possible configurations of L1-L2 (Spanish-English, English-Spanish). We noticed that in some cases same sentences are generated by different methods and sometimes there are no switches. We narrowed down the sentences by randomly choosing the combination of L1-L2 for each generator. This reduced the number of sentences from having 6 versions, to having only 3 versions of each sentence. From the resulting 30 sets, we removed 2 sets because one or more of the generator schemes produced a monolingual sentence. Therefore, we used 28 sets for human evaluations.

5.2 Human Evaluation Results

We had a total of 18 subjects participating in the experiment. All of them identified themselves as being able to read and write Spanish and English, and the majority of them said to have used CS at least some times. We showed to the human subjects the 28 sets of sentences. This time we included the original version of the sentence. Therefore, each judge was given 4 versions of each of the 28 code-switched sentences: the one generated from NB predictions, the one from VFI, the randomly generated, and the original one. Then we asked them to rate each sentence with a number from 1 to 5 indicating how natural and human-like the sentence sounds. A rating of 5 means that they strongly agree, 4 means they agree, 3 not sure, 2 disagree, 1 strongly disagree.

The average results are presented in Table 4. The sentences generated by NB were scored considerably higher than those from VFI and random, and closer to the human sentences. According to the paired t-test the difference between the NB score and the random one is significant (p=0.01). However the average score for VFI is lower than random. More experiments are needed to see if by choosing the setting where VFI had the highest F-measure would make a difference in this respect. Overall the subjects rated the human-generated CS sentences lower than what we were expecting, although it is clear that they consider these sentences more natural sounding than the rest. This low rating might be related to the attitude several evaluators expressed toward CS. In the evaluation form we asked the judges to express their opinion on CS and several of them indicated feelings along the lines of “we shouldn’t code-switch”.

There are several ways in which two parallel sentences can be combined in CS, and possibly several will sound natural, but from our results, it is clear that the NB algorithm was indeed able to generate a human-like CS behavior that was successfully differentiated from randomly-generated sentences.

By looking at the set of automatically generated code-switched sentences, we realized that the majority of the sentences are grammatical and natural sounding. We believe that for a large number of the sentences it would be hard for a human to distinguish the sentences that were automatically generated from the human-generated ones. One of the give away clues is when a multi-word expression is CS, or a tag line. Table 5 shows three examples from the sentences evaluated. In the table there is an example in sentence 1c where the noun phrase is code-switched, the sentence is grammatical according to Spanish rules, but it sounds very odd to have the noun *carta* followed by the adjective in English, “astrological”. Other interesting features are present in example 3 where for the same noun phrase “produce section” we have both, the female marking determiner *la* and the masculine *el*. The same thing happens for the noun phrase “check-out line”. We would need to have a larger occurrence of these instances in our test set to determine if on average one form is preferred over the other.

In another experiment, we measured the prediction performance of NB and VFI on the 30 code-switched sentences used in this part of the evaluation. The best results, an F-measure of 0.418, were achieved by NB when a context of 1 word was used, and no words, nor lemmas were included as features. This is the same setting used for the generation process. In contrast, VFI reached an F-measure of 0.351 on this same setting. 30 sentences represent a very small dataset but the results are very promising since the speakers are different in the training and testing dataset. Moreover, these results support the claim that written and spoken CS obey similar rules.

6 Related Work

There is little prior work on computational linguistic approaches to code-switched discourse. Most of the previous work includes formalisms to parsing and generating mixed sentences, for example for Marathi and English (Joshi, 1982), or Hindi and English (Goyal et al., 2003). Sankoff proposed a production model of bilingual discourse that accounts for the equivalence constraint and the unpredictability of code-switching (Sankoff, 1998a). His real-time production model draws on the alternation of fragments from two virtual monolingual sentences. But no statistical assessment has been conducted on real corpora.

Another related work deals with language identification on English-Maltese code-switched SMS messages (Rosner and Farrugia, 2007). What the authors found to work best for language identification in this noisy domain is a combination of a bigram Hidden Markov Model, trained on language transitions, and a trigram character Markov Model for handling unknown words.

1a. Naive Bayes:
By unlocking the information in your astrological chart, *puedo ver la respuesta!* Ask me!
1b. VFI:
Puedo ver la answer by unlocking the information in your *carta astrológica!* Ask me !
1c. Random:
By unlocking the information *de tu carta* astrological, I can see the answer! Ask me !
1d. Human:
By unlocking the information in your astrological chart, *puedo ver* the answer! *Pregúntame!*
1e. English version:
By unlocking the information in your astrological chart, I can see the answer! Ask me!
2a. Naive Bayes:
Pero siendo this a new year, *es tiempo de empezar de nuevo que no?*
2b. VFI:
But this being a new year, *es tiempo de empezar* over isn't it ?
2c. Random:
But this being a new *año*, it's *tiempo* to start over isn't it?
2d. Human:
Pero this being a new year, it's a time to start over *que no?*
2e. English version:
But this being a new year, it's time to start over isn't it?
3a. Naive Bayes:
Juan confirmed me that it was very obvious, *y no solamente en el* produce section, *en la* check-out line as well.
3b. VFI:
Me confirmó Juan que it was very obvious, *y no solamente en el* produce section, *también en la* check-out line.
3c. Random:
Juan confirmed *que fue* very obvious, *y* not *solamente en el área de* produce, in the check-out line as well.
3d. Human:
Me confirmó Juan que fue muy obvio, y no solamente en la produce section, *también en el* check-out line.
3e. English version:
Juan confirmed me that it was very obvious, and not only on the produce section, in the check-out line as well.

Table 5: Examples of automatically generated CS sentences.

7 Conclusions

We presented preliminary results on learning to predict CS points with machine learning. One of the possible applications of our method involves fine-tuning the weights in a multilingual language model, for instance, as part of a speech recognizer for Spanglish. With this in mind, we restricted the possible features in the learning scenario allowing only lexical and syntactic features that could be automatically generated from the text. Empirical evaluations on a Spanglish conversation showed that Naive Bayes and VFI can predict with acceptable F-measures possible CS points, considering the difficulty of the task. Prediction of CS points can help improve multilingual language models.

Evaluation of our approach cannot be done based only on the gold-standard set since there is no single right answer in this task. Therefore, we complemented the evaluation by involving judgements from bilingual speakers. We generated CS sentences by taking the predictions from the classifiers to merge parallel sentences. On average, the sentences generated from the NB model were rated closer to the original sentences, and a lot higher than the ones from a random generator. Most of the sentences sounded human-like. But because the process is automatic we did find some awkward constructions, for example plural vs singular noun-verb agreement, or multi-word phrases that were code-switched in the middle. Perhaps a multi-word recognition feature could improve results.

One of the advantages of technological development and economic globalization is that more people from different regions of the world with different cultures, and therefore, different languages will be in closer contact. As a result, code-switching will become more popular. It is important to start addressing this type of bilingual communication from a computational linguistics point of view. This work is one of the few attempts to fill the gap.

Some directions for future work include: exploring the extent to which our results can be improved by including a multi-word expression recognition system. We also want to investigate the integration of our approach to multilingual language models and move beyond CS to address other deeper linguistic phenomena. Lastly, we would like to explore similar approaches in other popular language combinations.

Acknowledgements

This research is supported by the National Science Foundation under grant 0812134. We are grateful to Ray Mooney, Melissa Sherman and the three anonymous reviewers for insightful comments and suggestions. Special thanks to the human judges that helped with the sentence evaluations.

References

G. Demiroz and H. A. Guvenir. 1997. Classification by voting feature intervals. In *European Conference on Machine Learning, ECML-97*, pages 85–92.

P. Goyal, Manav R. Mital, A. Mukerjee, Achla M. Raina, D. Sharma, P. Shukla, and K. Vikram. 2003. A bilingual parser for Hindi, English and code-switching structures. In *Computational Linguistics for South Asian Languages –Expanding Synergies with Europe, EACL-2003 Workshop*, Budapest, Hungary.

A. Joshi. 1982. Processing of sentences with intrasentential code-switching. In Ján Horecký, editor, *COLING-82*, pages 145–150, Prague, July.

J. Lipski. 1978. Code-switching and the problem of bilingual competence. In M. Paradis, editor, *Aspects of bilingualism*, pages 250–264. Hornbeam.

C. Montes-Alcalá. 2005. Mándame un e-mail: cambio de códigos español-inglés online. In Luis Ortiz and Manel Lacorte, editors, *In Contacto y contextos lingüísticos: El español en los Estados Unidos y en contacto con otras lenguas*. Iberoamericana/Vervuert.

C. Montes-Alcalá. 2007. Blogging in two languages: Code-switching in bilingual blogs. In Jonathan Holmquist, Augusto Lorenzino, and Lotfi Sayahi, editors, *In Selected Proc. of the Third Workshop on Spanish Sociolinguistics*, pages 162–170, Somerville, MA. Cascadilla Proceedings Project.

C. Myers-Scotton. 1997. *Duelling Languages: Grammatical Structure in Codeswitching*. Oxford University Press, 2nd edition.

A. Ng and M. Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and Naive Bayes. In *Advances in Neural Information Processing Systems (NIPS) 15*. MIT Press.

S. Poplack. 1980. Sometimes I'll start a sentence in Spanish y termino en español: toward a typology of code-switching. *Linguistics*, 18(7/8):581–618.

M. Rosner and P. J. Farrugia. 2007. A tagging algorithm for mixed language identification in a noisy domain. In *INTERSPEECH 2007*, pages 190–193, Antwerp, Belguim, August.

D. Sankoff. 1998a. A formal production-based explanation of the facts of code-switching. *Bilingualism, Language and Cognition*, (1):39–50.

D. Sankoff. 1998b. The production of code-mixed discourse. In *36th ACL*, volume I, pages 8–21, Montreal, Quebec, Canada, August.

H. Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*, September.

T. Solorio and Y. Liu. 2008. Part-of-speech tagging for English-Spanish code-switched text. In *EMNLP-2008*, Honolulu, Hawai, October.

I. H. Witten and E. Frank. 1999. *Data Mining, Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann.

Computing Word-Pair Antonymy

Saif Mohammad[†] **Bonnie Dorr**[*] **Graeme Hirst**[ϕ]

[†*]Laboratory for Computational Linguistics and Information Processing
[†*]Institute for Advanced Computer Studies and [*]Computer Science
[†*]University of Maryland and [*]Human Language Technology Center of Excellence
{saif,bonnie}@umiacs.umd.edu

[ϕ]Department of Computer Science
University of Toronto
gh@cs.toronto.edu

Abstract

Knowing the degree of antonymy between words has widespread applications in natural language processing. Manually-created lexicons have limited coverage and do not include most semantically contrasting word pairs. We present a new automatic and empirical measure of antonymy that combines corpus statistics with the structure of a published thesaurus. The approach is evaluated on a set of closest-opposite questions, obtaining a precision of over 80%. Along the way, we discuss what humans consider antonymous and how antonymy manifests itself in utterances.

1 Introduction

Native speakers of a language intuitively recognize different **degrees of antonymy**—whether two words are strongly antonymous (*hot–cold, good–bad, friend–enemy*), just semantically contrasting (*enemy–fan, cold–lukewarm, ascend–slip*) or not antonymous at all (*penguin–clown, cold–chilly, boat–rudder*). Over the years, many definitions of antonymy have been proposed by linguists (Cruse, 1986; Lehrer and Lehrer, 1982), cognitive scientists (Kagan, 1984), psycholinguists (Deese, 1965), and lexicographers (Egan, 1984), which differ from each other in small and large respects. In its strictest sense, antonymy applies to gradable adjectives, such as *hot–cold* and *tall–short*, where the two words represent the two ends of a semantic dimension. In a broader sense, it includes other adjectives, nouns, and verbs as well (*life–death, ascend–descend, shout–whisper*). In its broadest sense, it applies to any two words that represent contrasting meanings. We will use the term **degree of antonymy** to encompass the complete semantic range—a combined measure of the contrast in meaning conveyed by two words and the tendency of native speakers to call them opposites. The higher the degree of antonymy between a target word pair, the greater the semantic contrast between them and the greater their tendency to be considered antonym pairs by native speakers.

Automatically determining the degree of antonymy between words has many uses including detecting and generating paraphrases (*The dementors **caught** Sirius Black / Black could **not escape** the dementors*) and detecting contradictions (Marneffe et al., 2008; Voorhees, 2008) (*Kyoto has a predominantly **wet** climate / It is mostly **dry** in Kyoto*). Of course, such “contradictions” may be a result of differing sentiment, new information, non-coreferent mentions, or genuinely contradictory statements. Antonyms often indicate the discourse relation of contrast (Marcu and Echihabi, 2002). They are also useful for detecting humor (Mihalcea and Strapparava, 2005), as satire and jokes tend to have contradictions and oxymorons. Lastly, it is useful to know which words are semantically contrasting to a target word, even if simply to filter them out. For example, in the automatic creation of a thesaurus it is necessary to distinguish near-synonyms from word pairs that are semantically contrasting. Measures of distributional similarity fail to do so. Detecting antonymous words is not sufficient to solve most of these problems, but it remains a crucial, and largely unsolved, component.

Lexicons of pairs of words that native speakers consider antonyms have been created for certain languages, but their coverage has been limited. Further, as each term of an antonymous pair can have many semantically close terms, the contrasting word pairs far outnumber those that are commonly considered antonym pairs, and they remain unrecorded. Even though a number of computational approaches have been proposed for semantic closeness, and some for hypernymy–hyponymy (Hearst, 1992), measures of antonymy have been less successful. To some extent, this is because antonymy is not as well understood as other classical lexical-semantic relations.

We first very briefly summarize insights and intuitions about this phenomenon, as proposed by linguists and lexicographers (Section 2). We discuss related work (Section 3). We describe the resources we use (Section 4) and present experiments that examine the manifestation of antonymy in text (Sections 5 and 6). We then propose a new empirical approach to determine the degree of antonymy between two words (Section 7). We compiled a dataset of 950 closest-opposite questions, which we used for evaluation (Section 8). We conclude with a discussion of the merits and limitations of this approach and outline future work.

2 The paradoxes of antonymy

Antonymy, like synonymy and hyponymy, is a lexical-semantic relation that, strictly speaking, applies to two **lexical units**—combinations of surface form and word sense. (That said, for simplicity and where appropriate we will use the term "antonymous words" as a proxy for "antonymous lexical units".) However, accepting this leads to two interesting and seemingly paradoxical questions (described below in the two subsections).

2.1 Why are some pairs better antonyms?

Native speakers of a language consider certain contrasting word pairs to be antonymous (for example, *large–small*), and certain other seemingly equivalent word pairs as less so (for example, *large–little*). A number of reasons have been suggested: (1) Cruse (1986) observes that if the meaning of the target words is completely defined by one semantic dimension and the words represent the two ends of this semantic dimension, then they tend to be considered antonyms. We will refer to this semantic dimension as the **dimension of opposition**. (2) If on the other hand, as Lehrer and Lehrer (1982) point out, there is more to the meaning of the antonymous words than the dimension of opposition—for example, more semantic dimensions or added connotations—then the two words are not so strongly antonymous. Most people do not think of *chubby* as a direct antonym of *thin* because it has the additional connotation of being cute and informal. (3) Cruse (1986) also postulates that word pairs are not considered strictly antonymous if it is difficult to identify the dimension of opposition (for example, *city–farm*). (4) Charles and Miller (1989) claim that two contrasting words are identified as antonyms if they occur together in a sentence more often than chance. However, Murphy and Andrew (1993) claim that the greater-than-chance co-occurrence of antonyms in sentences is because together they convey contrast well, which is rhetorically useful, and not really the reason why they are considered antonyms in the first place.

2.2 Are semantic closeness and antonymy opposites?

Two words (more precisely, two lexical units) are considered to be close in meaning if there is a lexical-semantic relation between them. Lexical-semantic relations are of two kinds: **classical** and **non-classical**. Examples of classical relations include synonymy, hyponymy, troponymy, and meronymy. Non-classical relations, as pointed out by Morris and Hirst (2004), are much more common and include concepts pertaining to another concept (*kind, chivalrous, formal* pertaining to *gentlemanly*), and commonly co-occurring words (for example, problem–solution pairs such as *homeless, shelter*). Semantic distance (or closeness) in this broad sense is known as semantic relatedness. Two words are considered to be **semantically similar** if they are associated via the synonymy, hyponymy–hypernymy, or the troponymy relation. So terms that are semantically similar (*plane–glider, doctor–surgeon*) are also semantically related, but terms that are semantically related may not always be semantically similar (*plane–sky, surgeon–scalpel*).

Antonymy is unique among these relations because it simultaneously conveys both a sense of closeness and of distance (Cruse, 1986). Antonymous concepts are semantically related but not semantically similar.

3 Related work

Charles and Miller (1989) proposed that antonyms occur together in a sentence more often than chance. This is known as the **co-occurrence hypothesis**. They also showed that this was empirically true for four adjective antonym pairs. Justeson and Katz (1991) demonstrated the co-occurrence hypothesis for 35 prototypical antonym pairs (from an original set of 39 antonym pairs compiled by Deese (1965)) and also for an additional 22 frequent antonym pairs. All of these pairs were adjectives. Fellbaum (1995) conducted similar experiments on 47 noun, verb, adjective, and adverb pairs (noun–noun, noun–verb, noun–adjective, verb–adverb and so on) pertaining to 18 concepts (for example, *lose(v)–gain(n)* and *loss(n)–gain(n)*, where *lose(v)* and *loss(n)* pertain to the concept of "failing to have/maintain"). However, non-antonymous semantically related words such as hypernyms, holonyms, meronyms, and near-synonyms also tend to occur together more often than chance. Thus, separating antonyms from them has proven to be difficult.

Lin et al. (2003) used patterns such as "from *X* to *Y*" and "either *X* or *Y*" to separate antonym word pairs from distributionally similar pairs. They evaluated their method on 80 pairs of antonyms and 80 pairs of synonyms taken from the *Webster's Collegiate Thesaurus* (Kay, 1988). In this paper, we propose a method to determine the degree of antonymy between any word pair and not just those that are distributionally similar. Turney (2008) proposed a uniform method to solve word analogy problems that require identifying synonyms, antonyms, hypernyms, and other lexical-semantic relations between word pairs. However, the Turney method is supervised whereas the method proposed in this paper is completely unsupervised.

Harabagiu et al. (2006) detected antonyms for the purpose of identifying contradictions by using WordNet chains—synsets connected by the hypernymy–hyponymy links and exactly one antonymy link. Lucerto et al. (2002) proposed detecting antonym pairs using the number of words between two words in text and also cue words such as *but*, *from*, and *and*. Unfortunately, they evaluated their method on only 18 word pairs. Neither of these methods determines the degree of antonymy between words and they have not been shown to have substantial coverage. Schwab et al. (2002) create "antonymous vector" for a target word. The closer this vector is to the context vectors of the other target word, the more antonymous the two target words are. However, the antonymous vectors are manually created. Further, the approach is not evaluated beyond a handful of word pairs.

Work in sentiment detection and opinion mining aims at determining the polarity of words. For example, Pang, Lee and Vaithyanathan (2002) detect that adjectives such as *dazzling*, *brilliant*, and *gripping* cast their qualifying nouns positively whereas adjectives such as *bad*, *cliched*, and *boring* portray the noun negatively. Many of these gradable adjectives have antonyms. but these approaches do not attempt to determine pairs of positive and negative polarity words that are antonyms.

4 Resources

4.1 Published thesauri

Published thesauri, such as the *Roget's* and *Macquarie*, divide the vocabulary into about a thousand **categories**. Words within a category tend to be near-synonymous or semantically similar. One may also find antonymous and semantically related words in the same category, but this is rare. The intuition is that words within a category represent a coarse concept. Words with more than one meaning may be found in more than one category; these represent its coarse senses. Within a category, the words are grouped into **paragraphs**. Words in the same paragraph tend to be closer in meaning than those in different paragraphs. We will take advantage of the structure of the thesaurus in our approach.

4.2 WordNet

Unlike the traditional approach to antonymy, WordNet encodes antonymy as a lexical relationship—a relation between two words (not concepts) (Gross et al., 1989). Even though a synset (a WordNet concept) may be represented by more than one word, individual words across synsets are marked as (direct) antonyms. Gross et al. argue that other words in the synsets form "indirect antonyms".

Even after including the indirect antonyms, WordNet's coverage is limited. As Marcu and Echihabi (2002) point out, WordNet does not encode antonymy across part-of-speech (for example, *legally–embargo*). Further, the noun–noun, verb–verb, and adjective–adjective antonym pairs of WordNet largely ignore near-opposites as revealed by our experiments (Section 8 below). Also, WordNet (or any other manually-created repository of antonyms for that matter) does not encode the *degree* of antonymy between words. Nevertheless, we investigate the usefulness of WordNet as a source of seed antonym pairs for our approach.

4.3 Co-occurrence statistics

The **distributional hypothesis of closeness** states that words that occur in similar contexts tend to be semantically close (Firth, 1957). Distributional measures of distance, such as those proposed by Lin (1998), quantify how similar the two sets of contexts of a target word pair are. Equation 1 is a modified form of Lin's measure that ignores syntactic dependencies and hence it estimates semantic relatedness rather than semantic similarity:

$$Lin(w_1,w_2) = \frac{\sum_{w \in T(w_1) \cap T(w_2)} (I(w_1,w) + I(w_2,w))}{\sum_{w' \in T(w_1)} I(w_1,w') + \sum_{w'' \in T(w_2)} I(w_2,w'')} \quad (1)$$

Here w_1 and w_2 are the target words; $I(x,y)$ is the pointwise mutual information between x and y; and $T(x)$ is the set of all words y that have positive pointwise mutual information with the word x ($I(x,y) > 0$).

Mohammad and Hirst (2006) showed that these distributional word-distance measures perform poorly when compared with WordNet-based concept-distance measures. They argued that this is because the word-distance measures clump together the contexts of the different senses of the target words. They proposed a way to obtain distributional distance between word *senses*, using any of the distributional measures such as *cosine* or that proposed by Lin, and showed that this approach performed markedly better than the traditional *word-distance* approach. They used thesaurus categories as very coarse word senses. Equation 2 shows how Lin's formula is used to determine distributional distance between two thesaurus categories c_1 and c_2:

$$Lin(c_1,c_2) = \frac{\sum_{w \in T(c_1) \cap T(c_2)} (I(c_1,w) + I(c_2,w))}{\sum_{w' \in T(c_1)} I(c_1,w') + \sum_{w'' \in T(c_2)} I(c_2,w'')} \quad (2)$$

Here $T(c)$ is the set of all words w that have positive pointwise mutual information with the thesaurus category c ($I(c,w) > 0$). We adopt this method for use in our approach to determine word-pair antonymy.

5 The co-occurrence hypothesis of antonyms

As a first step towards formulating our approach, we investigated the co-occurrence hypothesis on a significantly larger set of antonym pairs than those studied before. We randomly selected a thousand antonym pairs (nouns, verbs, and adjectives) from WordNet and counted the number of times (1) they occurred individually and (2) they co-occurred in the same sentence within a window of five words, in the *British National Corpus (BNC)* (Burnard, 2000). We then calculated the mutual information for each of these word pairs and averaged it. We randomly generated another set of a thousand word pairs, without regard to whether they were antonymous or not, and used it as a control set. The average mutual information between the words in the antonym set was 0.94 with a standard deviation of 2.27. The average mutual information between the words in the control set was 0.01 with a standard deviation of 0.37. Thus antonymous word pairs occur together much more often than chance irrespective of their intended senses ($p < 0.01$). Of course, a number of non-antonymous words also tend to co-occur more often than chance—commonly known as collocations. Thus, strong co-occurrence is not a sufficient condition for detecting antonyms, but these results show that it can be a useful cue.

6 The substitutional and distributional hypotheses of antonyms

Charles and Miller (1989) also proposed that in most contexts, antonyms may be interchanged. The meaning of the utterance will be inverted, of course, but the sentence will remain grammatical and linguistically plausible. This came to be known as the **substitutability hypothesis**. However, their experiments did not support this claim. They found that given a sentence with the target adjective removed, most people did not confound the missing word with its antonym. Justeson and Katz (1991) later showed that in sentences that contain both members of an antonymous adjective pair, the target adjectives do indeed occur in similar syntactic structures at the phrasal level. From this (and to some extent from the co-occurrence hypothesis), we can derive the **distributional hypothesis of antonyms**: antonyms occur in similar contexts more often than non-antonymous words.

We used the same set of one thousand antonym pairs and one thousand control pairs as in the previous experiment to gather empirical proof of the distributional hypothesis. For each word pair from the antonym set, we calculated the distributional distance between each of their senses using Mohammad and Hirst's (2006) method of concept distance along with the modified form of Lin's (1998) distributional measure (equation 2). The distance between the closest senses of the word pairs was averaged for all thousand antonyms. The process was then repeated for the control set.

The control set had an average semantic closeness of 0.23 with a standard deviation of 0.11 on a scale from 0 (unrelated) to 1 (identical). On the other hand, antonymous word pairs had an average semantic closeness of 0.30 with a standard deviation of 0.23.[1] This demonstrates that relative to other word pairs, antonymous words tend to occur in similar contexts ($p < 0.01$). However, near-synonymous and similar word pairs also occur in similar contexts. (the distributional hypothesis of closeness). Thus, just like the co-occurrence hypothesis, occurrence in similar contexts is not sufficient, but rather yet another useful cue towards detecting antonyms.

[1]It should be noted that absolute values in the range between 0 and 1 are meaningless by themselves. However, if a set of word pairs is shown to consistently have higher values than another set, then we can conclude that the members of the former set tend to be semantically closer than those of the latter.

7 Our approach

We now present an empirical approach to determine the degree of antonymy between words. In order to maximize applicability and usefulness in natural language applications, we model the broad sense of antonymy. Given a target word pair, the approach determines whether they are antonymous or not, and if they are antonymous whether they have a high, medium, or low degree of antonymy. More precisely, the approach presents a way to determine whether one word pair is more antonymous than another.

The approach relies on the structure of the published thesaurus as well as the co-occurrence and distributional hypotheses. As mentioned earlier, a thesaurus organizes words in sets representing concepts or categories. We first determine pairs of thesaurus categories that are contrasting in meaning (Section 7.1). We then use the co-occurrence and distributional hypotheses to determine the degree of antonymy (Section 7.2).

7.1 Detecting contrasting categories

We propose two ways of detecting thesaurus category pairs that represent contrasting concepts (we will call these pairs **contrasting categories**): (1) using a seed set of antonyms and (2) using a simple heuristic that exploits how thesaurus categories are ordered.

7.1.1 Seed sets

Affix-generated seed set Antonym pairs such as *hot–cold* and *dark–light* occur frequently in text, but in terms of type-pairs they are outnumbered by those created using affixes, such as *un-* (*clear–unclear*) and *dis-* (*honest–dishonest*). Further, this phenomenon is observed in most languages (Lyons, 1977).

Table 1 lists sixteen morphological rules that tend to generate antonyms in English. These rules were applied to each of the words in the *Macquarie Thesaurus* and if the resulting term was also a valid word in the thesaurus, then the word-pair was added to the **affix-generated seed set**. These sixteen rules generated 2,734 word pairs. Of course, not all of them are antonymous, for example *sect–insect* and *coy–decoy*. However, these are relatively few in

w_1	w_2	**example pair**	w_1	w_2	**example pair**	w_1	w_2	**example pair**
X	*ab*X	*normal–abnormal*	X	*mis*X	*fortune–misfortune*	*im*X	*ex*X	*implicit–explicit*
X	*anti*X	*clockwise–anticlockwise*	X	*non*X	*aligned–nonaligned*	*in*X	*ex*X	*introvert–extrovert*
X	*dis*X	*interest–disinterest*	X	*un*X	*biased–unbiased*	*up*X	*down*X	*uphill–downhill*
X	*im*X	*possible–impossible*	*l*X	*ill*X	*legal–illegal*	*over*X	*under*X	*overdone–underdone*
X	*in*X	*consistent–inconsistent*	*r*X	*ir*X	*regular–irregular*	X*less*	X*ful*	*harmless–harmful*
X	*mal*X	*adroit–maladroit*						

Table 1: Sixteen affix rules to generate antonym pairs. Here 'X' stands for any sequence of letters common to both words w_1 and w_2.

number and were found to have only a small impact on the results.

WordNet seed set We compiled a list of 20,611 semantically contrasting word pairs from WordNet. If two words from two synsets in WordNet are connected by an antonymy link, then every possible word pair across the two synsets was considered to be semantically contrasting. A large number of them include multiword expressions. For only 10,807 of the 20,611 pairs were both words found in the *Macquarie Thesaurus*—the vocabulary used for our experiments. We will refer to them as the **WordNet seed set**.

Then, given these two seed sets, if any word in thesaurus category C_1 is antonymous to any word in category C_2 as per a seed antonym pair, then the two categories are marked as contrasting. It should be noted, however, that the seed antonym pair may be antonymous only in certain senses. For example, consider the antonym pair *work–play*. Here, *play* is antonymous to *work* only in its ACTIVITY FOR FUN sense and not its DRAMA sense. In such cases, we employ the distributional hypothesis of closeness: two words are antonymous to each other in those senses which are closest in meaning to each other. Since the thesaurus category pertaining to WORK is relatively closer in meaning to the ACTIVITY FOR FUN sense than the DRAMA sense, those two categories will be considered contrasting and not the categories pertaining to WORK and DRAMA.

If no word in C_1 is antonymous to any word in C_2, then the categories are considered not contrasting. As the seed sets, both automatically generated and manually created, are relatively large in comparison to the total number of categories in the *Macquarie Thesaurus* (812), this simple approach has reasonable coverage and accuracy.

7.1.2 Order of thesaurus categories

Most published thesauri are ordered such that contrasting categories tend to be adjacent. This is not a hard-and-fast rule, and often a category may be contrasting in meaning to several other categories. Further, often adjacent categories are not semantically contrasting. However, since this was an easy-enough heuristic to implement, we investigated the usefulness of considering adjacent categories as contrasting. We will refer to this as the **adjacency heuristic**.

7.2 Determining the degree of antonymy

Once we know which category pairs are contrasting (using the methods from the previous subsection), we determine the *degree* of antonymy between the two categories (Section 7.2.1). The aim is to assign contrasting category pairs a non-zero value signifying the degree of contrast. In turn, we will use that information to determine the degree of antonymy between any word pair whose members belong to two contrasting categories (Sections 7.2.2 and 7.2.3).

7.2.1 Category level

Using the distributional hypothesis of antonyms, we claim that the degree of antonymy between two *contrasting* concepts (thesaurus categories) is directly proportional to the distributional closeness of the two concepts. In other words, the more the words representing two contrasting concepts occur in similar contexts, the more the two concepts are considered to be antonymous.

Again we used Mohammad and Hirst's (2006) method along with Lin's (1998) distributional measure to determine the distributional closeness of two thesaurus concepts. Co-occurrence statistics required for the approach were computed from the

BNC. Words that occurred within a window of 5 words were considered to co-occur.

7.2.2 Lexical unit level

Recall that strictly speaking, antonymy (like other lexical-semantic relations) applies to lexical units (a combination of surface form and word sense). If two words are used in senses pertaining to contrasting categories (as per the methods described in Section 7.1), then we will consider them to be antonymous (degree of antonymy is greater than zero). If two words are used in senses pertaining to non-contrasting senses, then we will consider them to be not antonymous (degree of antonymy is equal to 0).

If the target words belong to the same thesaurus paragraphs as any of the seed antonyms linking the two contrasting categories, then the words are considered to have a high degree of antonymy. This is because words that occur in the same thesaurus paragraph tend to be semantically very close in meaning. Relying on the co-occurrence hypothesis, we claim that for word pairs listed in contrasting categories, the greater their tendency to co-occur in text, the higher their degree of antonymy. We use mutual information to capture the tendency of word–word co-occurrence.

If the target words do not both belong to the same paragraphs as a seed antonym pair, but occur in contrasting categories, then the target words are considered to have a low or medium degree of antonymy (less antonymous than the word pairs discussed above). Such word pairs that have a higher tendency to co-occur are considered to have a medium degree of antonymy, whereas those that have a lower tendency to co-occur are considered to have a low degree of antonymy.

Co-occurrence statistics for this purpose were collected from the *Google n-gram corpus* (Brants and Franz, 2006).[2] Words that occurred within a window of 5 words were considered to be co-occurring.

7.2.3 Word level

Even though antonymy applies to pairs of word and sense combinations, most available texts are not sense-annotated. If antonymous occurrences are to be exploited for any of the purposes listed in the beginning of this paper, then the text must be sense disambiguated. However, word sense disambiguation is a hard problem. Yet, and to some extent because unsupervised word sense disambiguation systems perform poorly, much can be gained by using simple heuristics. For example, it has been shown that cohesive text tends to have words that are close in meaning rather than unrelated words. This, along with the distributional hypothesis of antonyms, and the findings by Justeson and Katz (1991) (antonymous concepts tend to occur more often than chance in the same sentence), suggests that if we find a word pair in a sentence such that two of its senses are strongly contrasting (as per the algorithm described in Section 7.2.2), then it is probable that the two words are used in those contrasting senses.

8 Evaluation

8.1 Task and data

In order to best evaluate a computational measure of antonymy, we need a task that not only requires knowing whether two words are antonymous but also whether one word pair is more antonymous than another pair. Therefore, we evaluated our system on a set of closest-opposite questions. Each question has one target word and five alternatives. The objective is to identify that alternative which is the closest opposite of the target. For example, consider:

adulterate: a. *renounce* b. *forbid*
c. *purify* d. *criticize* e. *correct*

Here the target word is *adulterate*. One of the alternatives provided is *correct*, which as a verb has a meaning that contrasts with that of *adulterate*; however, *purify* has a greater degree of antonymy with *adulterate* than *correct* does and must be chosen in order for the instance to be marked as correctly answered. This evaluation is similar to how others have evaluated semantic distance algorithms on TOEFL synonym questions (Turney, 2001), except that in those cases the system had to choose the alternative which is *closest* in meaning to the target.

We looked on the World Wide Web for large sets of closest antonym questions. We found two independent sets of questions designed to prepare stu-

[2]We used the *Google n-gram corpus* is created from a text collection of over 1 trillion words. We intend to use the same corpus (and not the *BNC*) to determine semantic distance as well, in the near future.

	development data			**test data**		
	P	R	F	P	R	F
a. random baseline	0.20	0.20	0.20	0.20	0.20	0.20
b. affix-generated seeds only	0.72	0.53	0.61	0.71	0.51	0.60
c. WordNet seeds only	0.79	0.52	0.63	0.75	0.50	0.60
d. both seed sets	0.77	0.65	0.70	0.73	0.60	0.65
e. adjacency heuristic only	**0.81**	0.43	0.56	**0.83**	0.46	0.59
f. affix seed set + heuristic	0.75	0.60	0.67	0.76	0.61	0.68
g. both seed sets + heuristic	0.76	**0.66**	**0.70**	0.76	**0.64**	**0.70**

Table 2: Results obtained on closest-opposite questions.

dents for the Graduate Record Examination.[3] The first set consists of 162 questions. We used this set to develop our approach and will refer to it as the *development set*. Even though the algorithm does not have any tuned parameters per se, the development set helped determine which cues of antonymy were useful and which were not. The second set has 1208 closest-opposite questions. We discarded questions that had a multiword target or alternative. After removing duplicates we were left with 950 questions, which we used as the unseen *test set*.

Interestingly, the data contains many instances that have the same target word used in different senses. For example:

(1) ***obdurate:***	a. *meager*	b. *unsusceptible*
c. *right*	d. *tender*	e. *intelligent*
(2) ***obdurate:***	a. *yielding*	b. *motivated*
c. *moribund*	d. *azure*	e. *hard*
(3) ***obdurate:***	a. *transitory*	b. *commensurate*
c. *complaisant*	d. *similar*	e. *uncommunicative*

In (1), *obdurate* is used in the HARDENED IN FEELINGS sense and the closest opposite is *tender*. In (2), it is used in the RESISTANT TO PERSUASION sense and the closest opposite is *yielding*. In (3), it is used in the PERSISTENT sense and the closest opposite is *transitory*.

The datasets also contain questions in which one or more of the alternatives is a near-synonym of the target word. For example:

astute:	a. *shrewd*	b. *foolish*
c. *callow*	d. *winning*	e. *debating*

Observe that *shrewd* is a near-synonym of *astute*. The closest-opposite of *astute* is *foolish*. A manual check of a randomly selected set of 100 test-set questions revealed that, on overage, one in four had a near-synonym as one of the alternative.

[3]Both datasets are apparently in the public domain and will be made available on request.

8.2 Experiments

We used the algorithm proposed in Section 7 to automatically solve the closest-opposite questions. Since individual words may have more than one meaning, we relied on the hypothesis that the intended sense of the alternatives are those which are most antonymous to one of the senses of the target word. (This follows from the discussion earlier in Section 7.2.3.) So for each of the alternatives we used the target word as context (but not the other alternatives). We think that using a larger context to determine antonymy will be especially useful when the target words are found in sentences and natural text—something we intend to explore in the future.

Table 2 presents results obtained on the development and test data using different combinations of the seed sets and the adjacency heuristic. If the system did not find any evidence of antonymy between the target and any of its alternatives, then it refrained from attempting that question. We therefore report precision (number of questions answered correctly / number of questions attempted), recall (number of questions answered correctly / total number of questions), and F-score values ($2 \times P \times R/(P+R)$).

Observe that all results are well above the random baseline of 0.20 (obtained when a system randomly guesses one of the five alternatives to be the answer). Also, using only the small set of sixteen affix rules, the system performs almost as well as when it uses 10,807 WordNet antonym pairs. Using both the affix-generated and the WordNet seed sets, the system obtains markedly improved precision and coverage. Using only the adjacency heuristic gave best precision values (upwards of 0.8) with substantial coverage (attempting close to half the questions). However, best overall performance was obtained using both seed sets and the adjacency heuristic (F-score of 0.7).

8.3 Discussion

These results show that, to some degree, the automatic approach does indeed mimic human intuitions of antonymy. In tasks that require higher precision, using only the adjacency heuristic is best, whereas in tasks that require both precision and coverage, the seed sets may be included. Even when both seed sets were included, only four instances in the development set and twenty in the test set had target–answer pairs that matched a seed antonym pair. For all remaining instances, the approach had to generalize to determine the closest opposite. This also shows that even the seemingly large number of direct and indirect antonyms from WordNet (more than 10,000) are by themselves insufficient.

The comparable performance obtained using the affix rules alone suggests that even in languages without a wordnet, substantial accuracies may be achieved. Of course, improved results when using WordNet antonyms as well suggests that the information they provide is complementary.

Error analysis revealed that at times the system failed to identify that a category pertaining to the target word contrasted with a category pertaining to the answer. Additional methods to identify seed antonym pairs will help in such cases. Certain other errors occurred because one or more alternatives other than the official answer were also antonymous to the target. For example, the system chose *accept* as the opposite of *chasten* instead of *reward*.

9 Conclusion

We have proposed an empirical approach to antonymy that combines corpus co-occurrence statistics with the structure of a published thesaurus. The method can determine the degree of antonymy or contrast between any two thesaurus categories (sets of words representing a coarse concept) and between any two word pairs. We evaluated the approach on a large set of closest-opposite questions wherein the system not only identified whether two words are antonymous but also distinguished between pairs of antonymous words of different degrees. It achieved an F-score of 0.7 in this task where the random baseline was only 0.2. When aiming for high precision it scores over 0.8, but there is some drop in the number of questions attempted. In the process of developing this approach we validated the co-occurrence hypothesis proposed by Charles and Miller (1989) on a large set of 1000 noun, verb, and adjective pairs. We also gave empirical proof that antonym pairs tend to be used in similar contexts—the distributional hypothesis for antonyms.

Our future goals include porting this approach to a cross-lingual framework in order to determine antonymy in a resource-poor language by combining its text with a thesaurus from a resource-rich language. We will use antonym pairs to identify contrast relations between sentences to in turn improve automatic summarization. We also intend to use the approach proposed here in tasks where keyword matching is especially problematic, for example, separating paraphrases from contradictions.

Acknowledgments

We thank Smaranda Muresan, Siddharth Patwardhan, members of the CLIP lab at the University of Maryland, College Park, and the anonymous reviewers for their valuable feedback. This work was supported, in part, by the National Science Foundation under Grant No. IIS-0705832, in part, by the Human Language Technology Center of Excellence, and in part, by the Natural Sciences and Engineering Research Council of Canada. 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 sponsor.

References

Thorsten Brants and Alex Franz. 2006. Web 1t 5-gram version 1. *Linguistic Data Consortium*.

Lou Burnard. 2000. *Reference Guide for the British National Corpus (World Edition)*. Oxford University Computing Services.

Walter G. Charles and George A. Miller. 1989. Contexts of antonymous adjectives. *Applied Psychology*, 10:357–375.

David A. Cruse. 1986. *Lexical semantics*. Cambridge University Press.

James Deese. 1965. *The structure of associations in language and thought*. The Johns Hopkins Press.

Rose F. Egan. 1984. Survey of the history of English synonymy. *Webster's New Dictionary of Synonyms*, pages 5a–25a.

Christiane Fellbaum. 1995. Co-occurrence and antonymy. *International Journal of Lexicography*, 8:281–303.

John R. Firth. 1957. A synopsis of linguistic theory 1930–55. In *Studies in Linguistic Analysis*, pages 1–32, Oxford: The Philological Society. (Reprinted in F.R. Palmer (ed.), Selected Papers of J.R. Firth 1952-1959, Longman).

Derek Gross, Ute Fischer, and George A. Miller. 1989. Antonymy and the representation of adjectival meanings. *Memory and Language*, 28(1):92–106.

Sanda M. Harabagiu, Andrew Hickl, and Finley Lacatusu. 2006. Lacatusu: Negation, contrast and contradiction in text processing. In *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI-06)*, Boston, MA.

Marti Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the Fourteenth International Conference on Computational Linguistics*, pages 539–546, Nantes, France.

John S. Justeson and Slava M. Katz. 1991. Co-occurrences of antonymous adjectives and their contexts. *Computational Linguistics*, 17:1–19.

Jerome Kagan. 1984. *The Nature of the Child*. Basic Books.

Maire Weir Kay, editor. 1988. *Webster's Collegiate Thesaurus*. Merrian-Webster.

Adrienne Lehrer and K. Lehrer. 1982. Antonymy. *Linguistics and Philosophy*, 5:483–501.

Dekang Lin, Shaojun Zhao, Lijuan Qin, and Ming Zhou. 2003. Identifying synonyms among distributionally similar words. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 1492–1493, Acapulco, Mexico.

Dekang Lin. 1998. Automatic retreival and clustering of similar words. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING-98)*, pages 768–773, Montreal, Canada.

Cupertino Lucerto, David Pinto, and Héctor Jimiénez-Salazar. 2002. An automatic method to identify antonymy. In *Workshop on Lexical Resources and the Web for Word Sense Disambiguation*, pages 105–111, Puebla, Mexico.

John Lyons. 1977. *Semantics*, volume 1. Cambridge University Press.

Daniel Marcu and Abdesammad Echihabi. 2002. An unsupervised approach to recognizing discourse relations. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-02)*, Philadelphia, PA.

Marie-Catherine de Marneffe, Anna Rafferty, and Christopher D. Manning. 2008. Finding contradictions in text. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*, Columbus, OH.

Rada Mihalcea and Carlo Strapparava. 2005. Making computers laugh: Investigations in automatic humor recognition. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 531–538, Vancouver, Canada.

Saif Mohammad and Graeme Hirst. 2006. Distributional measures of concept-distance: A task-oriented evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Sydney, Australia.

Jane Morris and Graeme Hirst. 2004. Non-classical lexical semantic relations. In *Proceedings of the Workshop on Computational Lexical Semantics, HLT*, Boston, MA.

Gregory L. Murphy and Jane M. Andrew. 1993. The conceptual basis of antonymy and synonymy in adjectives. *Journal of Memory and Language*, 32(3):1–19.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 79–86, Philadelphia, PA.

Didier Schwab, Mathieu Lafourcade, and Violaine Prince. 2002. Antonymy and conceptual vectors. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING-02)*, pages 904–910.

Peter Turney. 2001. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the Twelfth European Conference on Machine Learning*, pages 491–502, Freiburg, Germany.

Peter Turney. 2008. A uniform approach to analogies, synonyms, antonyms, and associations. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING-08)*, pages 905–912, Manchester, UK.

Ellen M Voorhees. 2008. Contradictions and justifications: Extensions to the textual entailment task. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*, Columbus, OH.

Construction of an Idiom Corpus and its Application to Idiom Identification based on WSD incorporating Idiom-Specific Features

Chikara Hashimoto
Graduate School of Science and Engineering
Yamagata University
Yonezawa, Yamagata, 992-8510, JAPAN
ch@yz.yamagata-u.ac.jp

Daisuke Kawahara
National Institute of Information and
Communications Technology
Sorakugun, Kyoto, 619-0289, JAPAN
dk@nict.go.jp

Abstract

Some phrases can be interpreted either idiomatically (figuratively) or literally in context, and the precise identification of idioms is indispensable for full-fledged natural language processing (NLP). To this end, we have constructed an idiom corpus for Japanese. This paper reports on the corpus and the results of an idiom identification experiment using the corpus. The corpus targets 146 ambiguous idioms, and consists of 102,846 sentences, each of which is annotated with a literal/idiom label. For idiom identification, we targeted 90 out of the 146 idioms and adopted a word sense disambiguation (WSD) method using both common WSD features and idiom-specific features. The corpus and the experiment are the largest of their kind, as far as we know. As a result, we found that a standard supervised WSD method works well for the idiom identification and achieved an accuracy of 89.25% and 88.86% with/without idiom-specific features and that the most effective idiom-specific feature is the one involving the adjacency of idiom constituents.

1 Introduction

Some phrases like *kick the bucket* are ambiguous with regard to whether they carry literal or idiomatic meaning in a certain context. This ambiguity needs to be resolved in the same manner as ambiguous words that have been dealt with in the WSD literature. We term the resolution of the literal/idiomatic ambiguity as idiom identification, hereafter.

Idiom identification is classified into two kinds; one is for idiom types and the other is for idiom tokens. With the former, phrases that *can* be interpreted as idioms are found in text corpora, typically for compiling idiom dictionaries. On the other hand, the latter helps identify a phrase in context as a true idiom or a phrase that should be interpreted literally (a literal phrase, henceforth). In this paper, we deal with the latter, i.e., idiom token identification.

Despite the recent enthusiasm for multiword expressions (MWEs) (Grégoire et al., 2007; Grégoire et al., 2008), the idiom token identification is in an early phase of its development. Given that many NLP tasks like machine translation or parsing have been developed as a result of the availability of language resources, idiom token identification should also be developed when adequate idiom resources are provided. To this end, we have constructed a Japanese idiom corpus. We have also conducted an idiom identification experiment using the corpus that we hope will be a good reference point for future studies on the task. We drew on a standard WSD framework with machine learning exploiting both features commonly used in the WSD studies and idiom-specific features. This paper reports in detail the corpus and the result of the experiment; herein, it must be noted that to the best of our knowledge, the corpus and the experiment are the largest ever of their kind.

We only deal with the ambiguity between literal and idiomatic interpretations. However, some phrases have two or more idiomatic meanings without context. For example, a Japanese idiom *te-o dasu* (hand-ACC stretch)[1] can be interpreted as ei-

[1]ACC is the accusative case marker. Likewise we use the following notation in this paper; NOM for the nominative case

ther "punch," "steal" or "make moves on." This kind of ambiguity should be placed on the agenda.

We do not tackle the problem of what constitutes the notion of "idiom." We simply regard phrases listed in Sato (2007) as idioms.

The reminder of this paper is organized as follows. In §2 we present related works. §3 shows the target idioms. After the idiom corpus is described in §4, we detail our idiom identification method and experiment in §5. Finally §6 concludes the paper.

2 Related Work

There have only been a few works on the construction of an idiom corpus. In this regard, Birke and Sarkar (2006) and Cook et al. (2008) are notable exceptions. Birke and Sarkar (2006) automatically constructed a corpus of English idiomatic expressions (words that can be used non-literally). They targeted 50 expressions and collected about 6,600 examples. They call the corpus TroFi Example Base, which is available on the Web.[2] Cook et al. (2008) compiled a corpus of English verb-noun combinations (VNCs) tokens. Their corpus deals with 53 VNC expressions and consists of about 3,000 example sentences. Like ours, they assigned each example with a label indicating whether an expression in the example is used literally or idiomatically. Our corpus can be regarded as the Japanese idiom counterpart of these works. However, note that our corpus targets 146 idioms and consists of as many as 102,846 example sentences. Another exception is Tsuchiya et al. (2006), who manually constructed an example database of Japanese compound functional expressions named MUST. They provide it on the Web.[3] Some of the compound functional expressions in Japanese are ambiguous like idioms are.[4]

marker, DAT for the dative case marker, and GEN for the genitive case marker. FROM and TO stand for the Japanese counterparts of *from* and *to*. NEG represents a verbal negation morpheme.

[2]http://www.cs.sfu.ca/~anoop/students/jbirke/

[3]http://nlp.iit.tsukuba.ac.jp/must/

[4]For example, *(something)-ni-atatte* ((something)-DAT-run.into) means either "run into (something)" or "on the occasion of (something)." The former is the literal interpretation and the latter is the idiomatic interpretation of the compound functional expression.

The SAID dataset[5] provides data about the syntactic flexibility of English idioms. It does not concern itself with idiom token identification. However, as in Hashimoto et al. (2006b), Hashimoto et al. (2006a) and Cook et al. (2007) among others, the syntactic behavior of idioms is an important clue to idiom token identification.

Previous studies have mostly focused on the idiom type identification (Lin, 1999; Krenn and Evert, 2001; Baldwin et al., 2003; Shudo et al., 2004; Fazly and Stevenson, 2006). However, there has been a growing interest in idiom token identification in recent times (Katz and Giesbrecht, 2006; Hashimoto et al., 2006b; Hashimoto et al., 2006a; Birke and Sarkar, 2006; Cook et al., 2007). Katz and Giesbrecht (2006) compared the word vector of an idiom in context and that of the constituent words of the idiom using LSA in order to determine if the expression is idiomatic. Hashimoto et al. (2006b) and Hashimoto et al. (2006a) (HSU henceforth) focused their attention on the differences in grammatical constraints imposed on idioms and their literal counterparts such as the possibility of passivization, and developed handcrafted rules for Japanese idiom identification. Although their task is exactly the same as ours and we draw on the grammatical knowledge provided by them, the scale of their experiment is very small, since only 108 sentences were used for idiom identification in their paper. Further, unlike HSU, we employ matured WSD technologies. Cook et al. (2007) (CFS henceforth) propose an unsupervised method for English on the basis of the observation that idioms tend to be expressed in a small number of fixed forms.

These studies used only the characteristics of idioms (or MWEs). On the other hand, we exploit a WSD method, for which there have been many studies and matured technologies, in addition to the characteristics of idioms. Birke and Sarkar (2006) also used WSD. However, they employed an unsupervised method, while ours is a completely supervised one.

Apart from idioms, Uchiyama et al. (2005) conducted the token classification of Japanese compound verbs exploiting supervised method.

http://www.ldc.upenn.edu/Catalog/
[5]CatalogEntry.jsp?catalogId=LDC2003T10

3 Target Idioms

For this study, we selected 146 idioms through the following procedure. ① We extracted basic idioms from Sato (2007). Sato compiled about 3,600 basic idioms of Japanese from five books: two dictionaries for elementary school, two idiom dictionaries, and one linguistics book on idioms. We extracted those idioms that were described in more than two of these five books. The total number of such idioms added up to 926. ② From among these idioms, we chose ambiguous ones.[6] As a result, 146 idioms were selected.

As for ②, sometimes it is not trivial to determine if an idiom is ambiguous or not. Some idioms are rarely interpreted literally, while others, in all likelihood, take on the literal meaning. Is it meaningful to regard them as ambiguous and deal with them in this study? If not, how does one assuredly distinguish truly ambiguous idioms from those that are mostly interpreted either literally or figuratively? This can only be done if there is an accurate idiom identification system.

After all, we asked two native speakers of Japanese (Group A) to classify idioms into two classes: 1) truly ambiguous ones and 2) completely unambiguous or practically unambiguous ones. On the basis of the classification, one of the authors made final judgments.

To verify how stable this ambiguity endorsement was, we asked another two other native speakers of Japanese (Group B) to perform the same task and calculated the Kappa statistic between the two speakers. First, we sampled 101 idioms from the 926 chosen earlier. Then, the two members of Group B classified the sampled idioms into the two classes. The Kappa statistic was found to be 0.6576, which indicates middling stability.

Tables 2 and 3 list some of the target idioms.

4 Idiom Corpus

4.1 Corpus Specification

The corpus is designed for the idiom token identification task. That is, each example sentence in the corpus is annotated with a label that indicates whether the corresponding phrase in the example is used as an idiom or a literal phrase. We call the former the positive example and the latter the negative example. More specifically, the corpus consists of lines that each represent one example. A line consists of four fields as follows: $\boxed{1}$ **Label** indicates whether the example is positive or negative. Label *i* is used for positive examples and *l* for negative ones. $\boxed{2}$ **ID** denotes the idiom that is included in the example. In this study, each idiom has a unique number, which is based on Sato (2007). $\boxed{3}$ **Lemma** also shows the idiom in the example. We assigned each idiom its canonical (or standard) form on the basis of Sato (2007). $\boxed{4}$ **Example** is the example itself.

Given below is a sample of a negative example of *goma-o suru* (sesame-ACC crush) 'flatter'.

- *l*␣1417␣ごまをする␣すり鉢でごまをすり ···

The third field is the lemma of the idiom. The last one is the example that says 'crushing sesame in a mortar...'

Before working on the corpus construction, we prepared a reference by which human annotators could consistently distinguish between the literal and figurative meanings of idioms. To be more precise, this reference specified literal and idiomatic meanings for each idiom like dictionaries do. For example, the entry for *goma-o suru* in the reference is as follows.

Idiom:	To flatter people.
Literal:	To crush sesame.

As for the corpus size, we continued to annotate examples for each idiom, regardless of the proportion of idioms and literal phrases, until the total number of examples for each idiom reached 1,000.[7] In the case of a shortage of original data, we annotated as many examples as possible. The original data were sourced from the Japanese Web corpus (Kawahara and Kurohashi, 2006).

4.2 Corpus Construction

We constructed the corpus in the following manner: ❶ From the Web corpus, we collected example sentences that contained one of our target idioms whichever meaning (positive or negative) they

[6]Some idioms like *by and large* do not have a literal meaning. They are not dealt with in this paper.

[7]For idioms that we sampled for preliminary annotation, we annotated more than 1,000 examples.

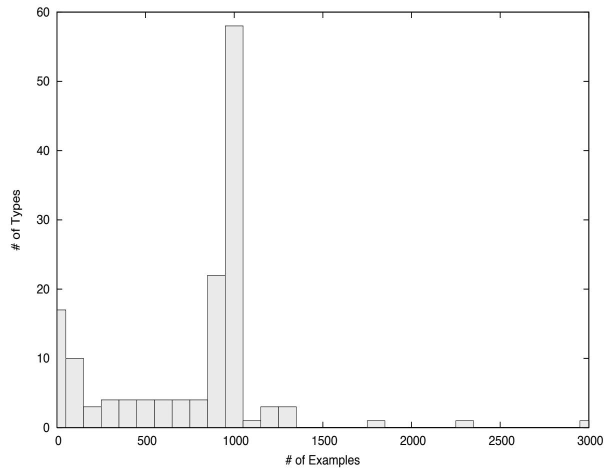


Figure 1: Distribution of the number of examples

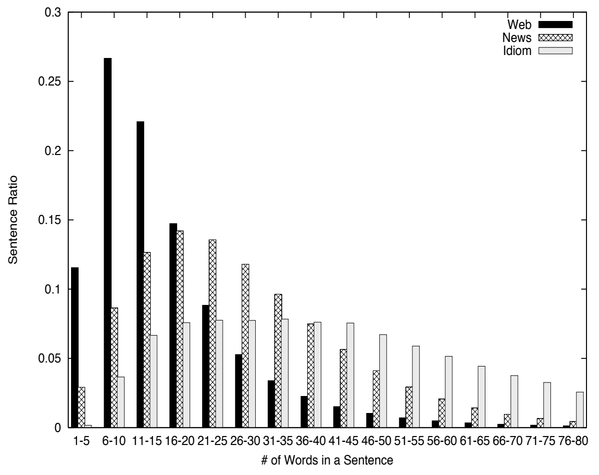


Figure 2: Distribution of sentence length

take on. Concretely speaking, we automatically collected sentences in which constituent words of one of our targets appeared in a canonical dependency relationship by using KNP[8], a Japanese dependency parser. ❷ We classified the collected examples as positive and negative. This was done by human annotators and was based on the reference to distinguish the two meanings. For annotation, longer examples were given higher priority than shorter examples. Note that we discarded examples that were collected by mistake due to dependency parsing errors and those that lacked a context that could help them be interpreted correctly.

This was done by the two members of Group A and took 230 hours.

4.3 Status of Corpus

The corpus consists of 102,846 examples.[9] Figure 1 shows the distribution of the number of examples. For 68 idioms, we annotated more than 1,000 examples. However, we annotated less than 100 examples for 17 idioms because of inadequate original data.

The average number of words in a sentence is 46. `Idiom` in Figure 2 shows the distribution of sentence length (the number of words) in the corpus. `Web` and `News` indicate the sentence length in the Web and a newspaper corpora, respectively. This is drawn from Kawahara and Kurohashi (2006). As you see, our corpus contains many more long sentences. This is because longer sentences were given priority for annotation, as stated in §4.2. Figure 3 shows the longest and shortest examples each for literal and idiomatic meanings of *goma-o suru* drawn from the corpus.

To determine how consistent the positive/negative annotation is across different human annotators, we sampled 1,421 examples from the corpus, asked the two members of Group B to do the same annotation, and calculated the Kappa statistic between the two. The value was 0.8519, which indicates very high agreement.

The corpus is available on the Web.[10] Currently we provide the list of the basic Japanese idioms we are dealing with, the idiom corpus, and the vector representation data used for the idiom identification experiment. The corpus is protected under the BSD license.

5 Idiom Identification Experiment

5.1 Method of Idiom Identification

We adopted a standard WSD method using machine learning. More specifically, we used SVM (Vapnik, 1995) with a quadratic kernel implemented in TinySVM.[11] The features we used are classified into either those that have been commonly used in WSD on the lines of Lee and Ng (2002) (LN hereafter),

[8]http://nlp.kuee.kyoto-u.ac.jp/nl-resource/knp.html

[9]Note that the figures reported here are for the corpus of the 2008-06-25 version and will be slightly changed over time.

[10]http://openmwe.sourceforge.jp/

[11]http://www.chasen.org/~taku/software/TinySVM/

- ただし、１５６２年にグレシャムは１５６２年に、同一の額面価値で流通する素材価値を異にする２種類の貨幣が存在すると劣悪な貨幣が流通に残り、優良な貨幣は駆逐されるという「グレシャムの法則」を発表していることから、女衒がしたたかであったように、ＩＴ興行師もメーカーに**ごまをすり**、政府の省庁にこびを売り、知性が無くても生命力だけで、目新しい言葉のつまみ食いで、悪毒、図々しく、虚勢で生き続けることだろう。
  (But I suspect that the show managers of IT ventures will remain sly and audacious, and survive by **flattering** manufacturers, bending over themselves to accede to the demands of governmental agencies, and talking glibly about buzz terms, without intelligence but with vitality, just like the brokers of prostitutes in the Edo period were, because Gresham's law of 1562 says that any circulating currency consisting of both good and bad money quickly becomes dominated by the bad money.)
- 上に**ごまをする**小役人タイプ。
  (Just like a pretty official **flattering** his boss.)

- 煮た大豆をつぶすには、ミンチみたいな器具があればいいのですが、ない　場合は、**ごまをする**もので潰すか、もっと簡単な方法としては、ビニール　袋に大豆を入れ、封をしてタオルをかけてその上から瓶でこするようにす　るといいでしょう。
  (In order to mash boiled soybeans, it is the best to use a meat chopper, but if you don't have one, use the thing to **crush sesame**, or put them into a plastic bag, cover it with a towel, then mash it with a glass bottle, which is easier.)
- **ごまをすり**調味料とあえる
  (**Crushing sesame**, then adding seasonings to it.)

Figure 3: The longest and shortest examples for both literal and idiomatic meanings of *goma-o suru*

or those that have been designed for Japanese idiom identification proposed by HSU.[12]

- Common WSD Features

  f1: POS of three words on the left side of idiom and three words on the right side

  f2: Local collocations

  f3: Single words in the surrounding context

  f4a: Lemma of the rightmost word among those words that are the dependents of the leftmost constituent word of idiom[13]

  f4b: POS of the rightmost word among those words that are the dependents of the leftmost constituent word of idiom

  f5a: Lemma of the word which the rightmost constituent word of idiom is the dependent of

  f5b: POS of the word which the rightmost constituent word of idiom is the dependent of

  f6: Hypernyms of words in the surrounding context

  f7: Domains of words (Hashimoto and Kurohashi, 2007; Hashimoto and Kurohashi, 2008) in the surrounding context

- Idiom-Specific Features

  f8: Adnominal modification flag

  f9: Topic case marking flag

  f10: Voice alternation flag

  f11: Negation flag

  f12: Volitional modality flag

  f13: Adjacency flag

We used JUMAN,[14] a morphological analyzer of Japanese, and KNP to extract these features.

[12]Remember that HSU implemented them in handcrafted rules. We adapted them to a machine learning framework.

[13]Note that Japanese is a head final language.

[14]`http://nlp.kuee.kyoto-u.ac.jp/nl-resource/juman.html`

f2 and **f3** are the same as those described in LN. But **f1** is slightly different in that we did not use the P_0 of LN. **f4** and **f5** roughly correspond to the syntactic relations of LN. We adapted it to Japanese idioms along with some simplifications. In the case of the example of *mune-o utu* (chest-ACC hit) 'impress' below,[15] **f4** is the POS and lemma of *tyousyu* and **f5** corresponds to those of *uta*.[16]

- tyousyu-no **mune-o** **utu** utukusi uta
  audience-GEN **chest-ACC** **hit** beautiful song
  'A beautiful song that **impresses** the audience'

f6 and **f7** are available from JUMAN's output. For example, the hypernym of *tyousyu* (audience) is `human` and its domain is `culture/media`. Those of *uta* (song) are `abstract-thing` and `culture/recreation`. They are not used in LN, but they are known to be useful for WSD (Tanaka et al., 2007; Magnini et al., 2002).

f8 indicates whether a nominal constituent of an idiom, if any, undergoes adnominal modification. **f9** indicates whether one of Japanese topic case markers is attached to a nominal constituent of an idiom, if any. **f10** is turned on when a passive or causative suffix is attached to a verbal constituent of an idiom, if any.[17] **f11** and **f12** are similar to **f10**. The former is used for negated forms and the latter for volitional modality suffixes of a predicate part of an idiom, if any.[18] Volitional modality includes expressions like order, request, permission, prohibition, and volition. Finally, **f13** indicates whether the constituents of an idiom is adjacent to each other.

As discussed in HSU, the idiom-specific features are effective to distinguish idioms from literal phrases. For example, the idiom *goma-o suru* does not allow adnominal modification, while its literal counterpart does. Similarly, the idiom *mune-o utu* cannot take volitional modality unlike its literal counterpart.

[15]The arrows indicate dependency relations.

[16]Functional words attaching to either the **f4** word or the **f5** word are ignored. In the example, *no* (GEN) is ignored.

[17]Passivization is indicated by the suffix *(r)are* in Japanese. But the same suffix is also used for honorification, potentials and spontaneous potentials. Since it is beyond the current technology, we gave up distinguishing them.

[18]Note that **f10**, **f11** and **f12** are applied to only those idioms that can be used as predicates.

5.2 Experimental Condition

In the experiment, we dealt with 90 idioms for which more than 50 examples for both idiomatic and literal usages were available.[19] We conducted experiments for each idiom.

The performance measure is the accuracy.

$$\text{Accuracy} = \frac{\text{\# of examples correctly identified}}{\text{\# of all example}}$$

The baseline system uniformly regards all examples as either positive or negative depending on which is more dominant in the idiom corpus. Naturally, this is prepared for each idiom.

$$\text{Baseline} = \frac{\max(\text{\# of positive, \# of negative})}{\text{\# of all example}}$$

The accuracy and the baseline accuracy for each idiom are calculated in a 10-fold cross validation style; we split examples of an idiom into 10 pieces in advance of the experiment.

Also, we calculated the overall accuracy and baseline accuracy from the individual results. We summed up all accuracy scores of all the 90 idioms and then divided it by 90, which is called the macro-average. We did this for the baseline accuracy, too.

Another performance measure is the relative error reduction (RER).[20]

$$\text{RER} = \frac{\text{ER of baseline} - \text{ER of system}}{\text{ER of baseline}}$$

The overall RER is calculated from the overall accuracy and baseline by the above formula.

5.3 Experimental Result

Table 1 shows the overall performance. The first column is the baseline accuracy (%). The second column is the accuracy (%) and relative error reduction (%) of the system without the idiom-specific features. The third column is those of the system with the idiom features. Tables 2 and 3 show the individual results of the 90 idioms. The first column shows

[19]Some examples were unavailable due to the feature extraction failure. Thus, examples used for the experiment are fewer in number than those included in the corpus.

[20]ER stands for Error Rate in the formula.

Table 2: Individual Results (1/2)

Type	Base	(Pos ; Neg)	w/o I (RER)	w/ I (RER)
青筋を立てる (blue.vein-ACC emerge) ‘burst a blood vessel’	83.38	(286 ; 57)	86.32 (17.68)	**86.61 (19.45)**
あぐらをかく (sit cross-legged) ‘rest on one’s laurels’	62.45	(587 ; 353)	92.66 (80.45)	**92.87 (81.02)**
足が付く (leg-NOM attach) ‘find a clue to solving a case’	72.21	(184 ; 478)	77.20 (17.96)	**79.62 (26.68)**
足が出る (leg-NOM go.out) ‘run over the budget’	77.59	(188 ; 651)	92.61 (67.01)	**93.08 (69.13)**
足元を見る (one’s feet-ACC look.down) ‘see someone coming’	57.53	(420 ; 310)	**85.89 (66.77)**	85.75 (66.45)
足を洗う (leg-ACC wash) ‘wash one’s hands of ...’	68.47	(632 ; 291)	92.65 (76.68)	92.65 (76.69)
足を伸ばす (leg-ACC stretch) ‘go a little further’	80.24	(727 ; 179)	95.26 (76.03)	**95.38 (76.59)**
頭が痛い (head-NOM ache) ‘harass oneself about ...’	57.87	(158 ; 217)	83.94 (61.89)	83.94 (61.89)
頭を抱える (head-ACC fold) ‘tear one’s hair out’	87.28	(796 ; 116)	91.35 (31.99)	91.35 (31.99)
頭をもたげる (head-ACC lift) ‘rear its head’	83.14	(804 ; 163)	93.40 (60.83)	**93.50 (61.45)**
脂が乗る (fat-NOM put.on) ‘warm up to one’s work’	83.69	(196 ; 1006)	92.94 (56.69)	92.94 (56.69)
油を売る (oil-ACC sell) ‘shoot the breeze’	86.67	(507 ; 78)	92.63 (44.70)	92.63 (44.70)
油を絞る (oil-ACC squeeze) ‘rake someone over the coals’	66.83	(69 ; 139)	84.64 (53.71)	**86.14 (58.23)**
網を張る (net-ACC spread) ‘wait expectantly’	70.10	(366 ; 858)	**81.28 (37.41)**	80.96 (36.31)
息が詰まる (breath-NOM choke.up) ‘stifling’	71.61	(681 ; 270)	**79.82 (28.91)**	79.50 (27.80)
一から十まで (one-FROM ten-TO) ‘all without exception’	92.00	(770 ; 67)	93.48 (18.51)	93.48 (18.51)
色を失う (color-ACC lose) ‘turn pale’	73.32	(262 ; 720)	84.23 (40.91)	84.23 (40.91)
腕が上がる (arm-NOM go.up) ‘develop one’s skill’	57.06	(481 ; 362)	84.47 (63.85)	**88.75 (73.80)**
尾を引く (tail-ACC pull) ‘have a lasting effect’	87.72	(843 ; 118)	93.14 (44.15)	**93.35 (45.84)**
顔を出す (face-ACC present) ‘show up’	84.48	(697 ; 128)	88.60 (26.49)	**88.82 (27.93)**
肩を並べる (shoulder-ACC juxtapose) ‘on a par’	89.38	(842 ; 100)	**93.20 (35.97)**	93.10 (34.97)
角が取れる (corner-NOM remove) ‘become mature’	57.45	(370 ; 274)	**78.35 (49.13)**	78.04 (48.39)
唇をかむ (lip-ACC bite) ‘bite one’s lip’	70.89	(587 ; 241)	78.40 (25.78)	**79.36 (29.10)**
口を切る (mouth-ACC cut) ‘break the ice’	51.50	(210 ; 223)	**84.83 (68.73)**	83.69 (66.36)
口をとがらせる (mouth-ACC sharpen) ‘pout’	86.33	(663 ; 105)	**87.61 (9.40)**	87.35 (7.47)
首が回らない (neck-NOM turn-NEG) ‘up to one’s neck’	66.63	(619 ; 310)	**86.41 (59.28)**	86.22 (58.71)
首を切る (neck-ACC cut) ‘give the axe’	53.90	(449 ; 384)	**89.93 (78.15)**	89.80 (77.88)
首をひねる (neck-ACC twist) ‘think hard’	93.16	(885 ; 65)	**94.11 (13.85)**	93.79 (9.23)
事によると (thing-DAT depend) ‘perhaps’	67.15	(231 ; 113)	96.50 (89.35)	**97.35 (91.94)**
ごまをする (sesame-ACC crush) ‘flatter’	50.29	(87 ; 88)	**92.75 (85.42)**	90.99 (81.88)
背を向ける (back-ACC train) ‘turn one’s back’	66.70	(597 ; 298)	89.06 (67.14)	89.06 (67.14)
血が通う (blood-NOM flow) ‘humane’	50.18	(422 ; 419)	82.41 (64.70)	**83.24 (66.37)**
宙に浮く (midair-DAT float) ‘’	58.07	(382 ; 529)	88.03 (71.46)	**88.69 (73.03)**
土が付く (dirt-NOM attach) ‘be defeated in sumo wrestling’	72.66	(70 ; 186)	**79.48 (24.97)**	78.76 (22.33)
手が届く (hand-NOM reach) ‘afford’ ‘reach an age’ ‘attentive’	80.76	(470 ; 112)	87.66 (35.85)	87.66 (35.85)
手がない (hand-NOM there.isn’t) ‘have no remedy’	86.94	(799 ; 120)	92.61 (43.38)	**92.83 (45.06)**
手が離れる (hand-NOM get.away) ‘get one’s work done’	53.49	(360 ; 414)	**92.37 (83.59)**	92.36 (83.57)
手に乗る (hand-DAT ride) ‘fall into someone’s trap’	61.05	(372 ; 583)	92.86 (81.68)	**93.49 (83.30)**
手を入れる (hand-DAT insert) ‘obtain’	53.21	(373 ; 328)	93.44 (85.99)	**93.59 (86.29)**
手を掛ける (hand-ACC hang) ‘give a lot of care’	70.57	(241 ; 578)	91.19 (70.04)	**91.31 (70.46)**
手を切る (hand-ACC cut) ‘break away’	57.85	(468 ; 341)	91.08 (78.83)	91.08 (78.83)
手を取る (hand-ACC take) ‘give every possible help (to learn)’	88.89	(91 ; 728)	**92.74 (34.67)**	92.62 (33.56)
手を握る (hand-ACC grasp) ‘conclude an alliance’	90.51	(73 ; 696)	**95.44 (51.93)**	95.17 (49.16)
手を延ばす (hand-ACC stretch) ‘extend one’s business’	89.55	(95 ; 814)	94.01 (42.69)	**94.22 (44.72)**
手を広げる (hand-ACC open.up) ‘extend one’s business’	70.52	(579 ; 242)	89.17 (63.26)	**90.15 (66.57)**
手を回す (hand-ACC turn) ‘take measures’	68.86	(246 ; 544)	93.04 (77.64)	**93.92 (80.49)**
峠を越す (mountain.pass-ACC go.over) ‘get over the hump’	72.18	(685 ; 264)	89.28 (61.46)	**89.49 (62.23)**
泥を塗る (mud-ACC daub) ‘drag someone through mud’	74.38	(543 ; 187)	91.64 (67.38)	**91.92 (68.45)**
波に乗る (wave-DAT ride) ‘catch a wave’	86.23	(783 ; 125)	**93.05 (49.55)**	92.94 (48.74)
熱が冷める (heat-NOM get.cool) ‘fever goes down’	89.90	(890 ; 100)	92.02 (21.00)	**92.22 (23.00)**
熱を上げる (heat-ACC raise) ‘go ape’	92.52	(903 ; 73)	94.50 (26.45)	**94.71 (29.21)**
熱を入れる (heat-ACC feed.in) ‘enthuse’	85.06	(723 ; 127)	90.71 (37.80)	**91.76 (44.88)**
根を下ろす (root-ACC take.down) ‘take root’	85.83	(824 ; 136)	93.23 (52.21)	93.23 (52.21)
根を張る (root-ACC spread) ‘take root’	60.00	(564 ; 376)	87.66 (69.15)	87.66 (69.15)
バスに乗り遅れる (bus-DAT miss) ‘miss the boat’	76.97	(199 ; 665)	90.50 (58.74)	**92.36 (66.81)**
バトンを渡す (baton-ACC give) ‘have someone succeed his position’	65.33	(471 ; 250)	81.70 (47.23)	**82.25 (48.81)**
鼻息が荒い (nasal.breathing-NOM heavy) ‘full of big talk’	52.77	(286 ; 256)	75.33 (47.77)	**76.62 (50.50)**
鼻が高い (nose-NOM high) ‘proud’	50.27	(659 ; 652)	81.01 (61.81)	**82.30 (64.42)**
鼻を折る (nose-ACC break) ‘humble (someone)’	56.60	(69 ; 90)	69.58 (29.91)	**74.92 (42.20)**
鼻を鳴らす (nose-ACC make.a.sound) ‘make light of ...’	55.72	(536 ; 426)	80.79 (56.63)	**81.21 (57.57)**
腹を割る (belly-ACC cut) ‘have a heart-to-heart talk’	95.62	(1265 ; 58)	96.68 (24.16)	96.68 (24.16)
歯を食い縛る (teeth-ACC clench) ‘grit one’s teeth’	65.54	(194 ; 102)	**71.97 (18.66)**	71.63 (17.66)
人を食う (human-ACC eat) ‘look down on someone’	74.95	(727 ; 243)	87.01 (48.15)	87.01 (48.15)
火花を散らす (spark-ACC spread) ‘fight heatedly’	75.99	(728 ; 230)	89.57 (56.56)	**89.68 (57.00)**

Table 3: Individual Results (2/2)

Type	Base	(Pos ; Neg)	w/o I (RER)	w/ I (RER)
筆を入れる (painting.brush-ACC add) ‘correct (writings or paintings)’	75.80	(213 ; 68)	83.99 (33.84)	**84.70 (36.79)**
船をこぐ (ship-ACC row) ‘nod’	50.76	(167 ; 162)	75.82 (50.88)	**76.37 (52.01)**
骨が折れる (bone-NOM break) ‘have difficulty’	62.30	(575 ; 348)	94.14 (84.46)	94.14 (84.47)
骨を埋める (bone-ACC bury) ‘make it one’s final home’	82.82	(757 ; 157)	89.84 (40.85)	**90.60 (45.31)**
骨を折る (bone-ACC break) ‘make efforts’	60.89	(350 ; 545)	92.74 (81.43)	**92.96 (82.01)**
幕が開く (curtain-NOM open) ‘start’	55.64	(533 ; 425)	**86.32 (69.17)**	86.22 (68.94)
右から左 (right-FROM left) ‘passing through without staying’	73.88	(794 ; 2246)	**89.90 (61.34)**	89.87 (61.21)
水と油 (water-AND oil) ‘oil and water’	55.66	(1053 ; 839)	83.19 (62.10)	**85.84 (68.07)**
水に流す (water-DAT flush) ‘forgive and forget’	67.08	(652 ; 320)	85.91 (57.19)	**89.40 (67.81)**
身に付ける (body-DAT put.on) ‘learn’	90.29	(725 ; 78)	**96.51 (64.11)**	96.39 (62.82)
耳が痛い (ear-NOM ache) ‘make one’s ears burn’	59.49	(333 ; 489)	88.69 (72.08)	**89.54 (74.19)**
耳に入れる (ear-DAT insert) ‘get word of ...’	74.89	(501 ; 168)	89.50 (58.20)	**90.38 (61.67)**
実を結ぶ (fruit-ACC bear) ‘bear fruit’	89.39	(826 ; 98)	**95.79 (60.33)**	95.68 (59.31)
胸が痛む (chest-NOM ache) ‘suffer heartache’	93.59	(876 ; 60)	95.82 (34.78)	**95.93 (36.46)**
胸が膨らむ (chest-NOM expand) ‘feel one’s heart leap’	55.58	(338 ; 423)	94.08 (86.68)	**94.48 (87.57)**
胸を打つ (chest-ACC hit) ‘impress’	92.39	(801 ; 66)	96.45 (53.34)	**96.68 (56.39)**
芽が出る (germ-NOM come.out) ‘close to making the top’	56.57	(377 ; 491)	91.33 (80.03)	**91.55 (80.55)**
目がない (eye-NOMthere.isn’t) ‘have a passion for ...’	91.81	(829 ; 74)	**95.70 (47.47)**	95.25 (42.05)
メスを入れる (scalpel-ACC insert) ‘take drastic measures’	88.96	(741 ; 92)	96.28 (66.30)	96.28 (66.30)
目に入る (eye-DAT enter) ‘catch sight of ...’	84.76	(623 ; 112)	90.22 (35.79)	**91.16 (41.97)**
目を覆う (eye-ACC cover) ‘be in a shambles’	87.24	(725 ; 106)	91.45 (32.99)	**92.06 (37.72)**
目を覚ます (eye-ACC awake) ‘snap out of ..’	83.26	(118 ; 587)	87.92 (27.85)	**88.64 (32.12)**
目をつぶる (eye-ACC close) ‘turn a blind eye’	70.13	(533 ; 227)	90.26 (67.40)	90.26 (67.40)
目を細くする (eye-ACC thin) ‘one’s eyes light up’	53.44	(115 ; 132)	**75.20 (46.74)**	75.11 (46.54)
指をくわえる (finger-ACC suck) ‘look enviously’	92.50	(876 ; 71)	**95.68 (42.41)**	95.58 (41.09)
弓を引く (bow-ACC draw) ‘defy’	88.06	(138 ; 1018)	**95.51 (62.41)**	95.43 (61.68)

Table 1: Overall Result

Base	w/o I (RER)	w/ I (RER)
72.92	88.86 (58.87)	89.25 (60.30)

the target idioms. The second column shows baseline accuracy (%) and the numbers of positive and negative examples for each idiom. The accuracy (%) and relative error reduction (%) of the system without the idiom-specific features are described in the third column. The fourth column is those of the system with the idiom features. Bold face indicates a better performance.

All in all, we see relatively high baseline performances. Nevertheless, both systems outperformed the baseline. Especially, the system without the idiom-specific features has a noticeable lead over the baseline, showing that WSD technologies are effective in the idiom identification. Incorporating the idiom features into the system improved the overall performance, which is statistically significant (McNemar test, $p<0.01$). But performances of some idioms slightly degraded by the incorporation of the idiom features.

Table 4: Overall Results without Using One of the Idiom Features

Feature Type	Acc
All	89.25
−f8 (w/o Adnominal modification flag)	89.24
−f9 (w/o Topic case marking flag)	89.22
−f10 (w/o Voice alternation flag)	89.15
−f11 (w/o Negation flag)	89.17
−f12 (w/o Volitional modality flag)	89.19
−f13 (w/o Adjacency flag)	89.09

Table 4 shows overall results without using one of the idiom features.[21] As you see, the adjacency flag (f13) contributes to idiom identification accuracy the most.[22] On the other hand, the adnominal modification flag (f8) contributes to the task only slightly.[23]

[21]The first row shows the result with all idiom features used, just for ease of reference.

[22]Note that greater performance drop indicates greater contribution.

[23]This result is inconsistent with the result obtained in HSU, where they reported that grammatical constraints involving adnominal modification was most effective. This inconsistency might be attributed to the differences of datasets being used for idiom identification experiment. HSU used only 108 sentences

Table 5: Results reported in CFS

	Accu	RER
Baseline	61.9	—
Unsupervised	72.4	27.6
Supervised	76.2	37.5

Table 5 shows the results reported in CFS. Their baseline system regards all instances as idioms. The performance of the supervised one is obtained by the method of Katz and Giesbrecht (2006). Though we cannot simply compare this with our results due to the difference in experimental conditions, this implies that our WSD-based method was equally good or possibly better than their methods that are tailored to MWEs.

6 Conclusion

In this paper, we reported on the idiom corpus we have constructed and the idiom identification experiment using the corpus.

As mentioned in §4.3, some idioms are short of examples in the current idiom corpus. We plan to collect more examples by using different characters. In the Japanese language, there are basically three character systems: Hiragana, Katakana, and Chinese characters. Thus, you can write an idiom in different characters. For example, *mune-o utu* (chest-ACC hit) 'impress' can be either 胸を打つ or 胸をうつ.

In spite of its imperfection, we are sure that we can learn a lot about the idiom identification from the corpus, since, as far as we know, it is the largest-ever one, and so is the idiom identification experiment reported in §5.

Also, we showed that a standard supervised WSD method works well for the idiom identification. Our system achieved the accuracy of 89.25% and 88.86% with/without idiom-specific features.

Though we dealt with as many as 90 idioms, practical NLP systems are required to deal with many more idioms. Toward a scalable idiom identification, we have to develop an unsupervised or semi-supervised method. The unsupervised method of Birke and Sarkar (2006) requires WordNet. Fortunately, the Japanese WordNet is now available (Isahara et al., 2008), thus we can try their method. Also, CFS propose a language-independent unsupervised method. These could be of help.

At any rate, our idiom corpus will play an important role in the development of unsupervised or semi-supervised methods, and the experimental results obtained in this study will be a good reference point to evaluate those methods.

for the experiment, while 75,011 sentences were used for our experiment. Also, the dataset of HSU came from newspaper articles, while our dataset came from the web.

Acknowledgments

This work was conducted as a part of the collaborative research project of Kyoto University and NTT Communication Science Laboratories.

The work was supported from NTT Communication Science Laboratories and JSPS Grants-in-Aid for Young Scientists (B) 19700141.

We would like to thank the members of the collaborative research group of Kyoto University and NTT Communication Science Laboratories and Francis Bond for their stimulating discussion. Our thanks go as well to Prof. Sato Satoshi, who kindly gave us the list of basic idioms of Japanese.

References

Timothy Baldwin, Colin Bannard, Takaaki Tanaka, and Dominic Widdows. 2003. An empirical model of multiword expression decomposability. In *Proceedings of the ACL 2003 workshop on Multiword expressions*, pages 89–96.

Julia Birke and Anoop Sarkar. 2006. A clustering approach for the nearly unsupervised recoginition of nonliteral language. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2006)*, pages 329–336.

Paul Cook, Afsaneh Fazly, and Suzanne Stevenson. 2007. Pulling their weight: Exploiting syntactic forms for the automatic identification of idiomatic expressions in context. In *Proceedings of the ACL 2007 Workshop on A Broader Perspective on Multiword Expressions*, pages 41–48.

Paul Cook, Afsaneh Fazly, and Suzanne Stevenson. 2008. The VNC-Tokens Dataset. In *Proceedings of the LREC Workshop Towards a Shared Task for Multiword Expressions (MWE2008)*, pages 19–22.

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

Nicole Grégoire, Stefan Evert, and Su Nam Kim, editors. 2007. *Proceedings of the Workshop on A Broader Perspective on Multiword Expressions*. Association for Computational Linguistics, Prague.

Nicole Grégoire, Stefan Evert, and Brigitte Krenn, editors. 2008. *Proceedings of the LREC Workshop Towards a Shared Task for Multiword Expressions*. ACL Special Interest Group on the Lexicon (SIGLEX), Marrakech.

Chikara Hashimoto and Sadao Kurohashi. 2007. Construction of Domain Dictionary for Fundamental Vocabulary. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL'07) Poster*, pages 137–140.

Chikara Hashimoto and Sadao Kurohashi. 2008. Blog Categorization Exploiting Domain Dictionary and Dynamically Estimated Domains of Unknown Words. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL'08) Short paper, Poster*, pages 69–72.

Chikara Hashimoto, Satoshi Sato, and Takehito Utsuro. 2006a. Detecting Japanese idioms with a linguistically rich dictionary. *Language Resources and Evaluation*, 40(3–4):243–252.

Chikara Hashimoto, Satoshi Sato, and Takehito Utsuro. 2006b. Japanese Idiom Recognition: Drawing a Line between Literal and Idiomatic Meanings. In *The Joint 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL 2006) Poster*, pages 353–360, Sydney, July.

Hitoshi Isahara, Francis Bond, Kiyotaka Uchimoto, Masao Utiyama, and Kyoko Kanzaki. 2008. Development of the Japanese WordNet. In *The sixth international conference on Language Resources and Evaluation (LREC2008)*.

Graham Katz and Eugenie Giesbrecht. 2006. Automatic identification of non-compositional multi-word expressions using latent semantic analysis. In *Proceedings of the Workshop, COLING/ACL 2006, Multiword Expressions: Identifying and Exploiting Underlying Properties*, pages 12–19, July.

Daisuke Kawahara and Sadao Kurohashi. 2006. Case Frame Compilation from the Web using High-Performance Computing. In *Proceedings of The 5th International Conference on Language Resources and Evaluation (LREC-06)*, pages 1344–1347.

Brigitte Krenn and Stefan Evert. 2001. Can we do better than frequency? a case study on extracting pp-verb collocations. In *Proceedings of the ACL-01 Workshop on Collocations*, pages 39–46.

Yoong Keok Lee and Hwee Tou Ng. 2002. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 41–48.

Dekang Lin. 1999. Automatic identification of non-compositional phrases. In *Proceeding of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 317–324.

Bernardo Magnini, Carlo Strapparava, Giovanni Pezzulo, and Alfio Gliozzo. 2002. The Role of Domain Information in Word Sense Disambiguation. *Natural Language Engineering, special issue on Word Sense Disambiguation*, 8(3):359–373.

Satoshi Sato. 2007. Compilation of a comparative list of basic Japanese idioms from five sources. In *IPSJ 2007-NL-178*, pages 1–6. (in Japanese).

Kosho Shudo, Toshifumi Tanabe, Masahito Takahashi, and Kenji Yoshimura. 2004. MWEs as Non-propositional Content Indicators. In *the 2nd ACL Workshop on Multiword Expressions: Integrating Processing*, pages 32–39.

Takaaki Tanaka, Francis Bond, Timothy Baldwin, Sanae Fujita, and Chikara Hashimoto. 2007. Word Sense Disambiguation Incorporating Lexical and Structural Semantic Information. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 477–485.

Masatoshi Tsuchiya, Takehito Utsuro, Suguru Matsuyoshi, Satoshi Sato, and Seiichi Nakagawa. 2006. Development and analysis of an example database of Japanese compound functional expressions. *Transactions of Information Processing Society of Japan*, 47(6):1728–1741. (in Japanese).

Kiyoko Uchiyama, Timothy Baldwin, and Shun Ishizaki. 2005. Disambiguating Japanese compound verbs. *Computer Speech and Language, Special Issue on Multiword Expressions*, 19(4):497–512.

Vladimir Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.

Word Sense Disambiguation Using OntoNotes: An Empirical Study

Zhi Zhong and **Hwee Tou Ng** and **Yee Seng Chan**
Department of Computer Science
National University of Singapore
Law Link, Singapore 117590
{zhongzhi, nght, chanys}@comp.nus.edu.sg

Abstract

The accuracy of current word sense disambiguation (WSD) systems is affected by the fine-grained sense inventory of WordNet as well as a lack of training examples. Using the WSD examples provided through OntoNotes, we conduct the first large-scale WSD evaluation involving hundreds of word types and tens of thousands of sense-tagged examples, while adopting a coarse-grained sense inventory. We show that though WSD systems trained with a large number of examples can obtain a high level of accuracy, they nevertheless suffer a substantial drop in accuracy when applied to a different domain. To address this issue, we propose combining a domain adaptation technique using feature augmentation with active learning. Our results show that this approach is effective in reducing the annotation effort required to adapt a WSD system to a new domain. Finally, we propose that one can maximize the dual benefits of reducing the annotation effort while ensuring an increase in WSD accuracy, by only performing active learning on the set of most frequently occurring word types.

1 Introduction

In language, many words have multiple meanings. The process of identifying the correct meaning, or sense of a word in context, is known as word sense disambiguation (WSD). WSD is one of the fundamental problems in natural language processing and is important for applications such as machine translation (MT) (Chan et al., 2007a; Carpuat and Wu, 2007), information retrieval (IR), etc.

WSD is typically viewed as a classification problem where each ambiguous word is assigned a sense label (from a pre-defined sense inventory) during the disambiguation process. In current WSD research, WordNet (Miller, 1990) is usually used as the sense inventory. WordNet, however, adopts a very fine level of sense granularity, thus restricting the accuracy of WSD systems. Also, current state-of-the-art WSD systems are based on supervised learning and face a general lack of training data.

To provide a standardized test-bed for evaluation of WSD systems, a series of evaluation exercises called SENSEVAL were held. In the English all-words task of SENSEVAL-2 and SENSEVAL-3 (Palmer et al., 2001; Snyder and Palmer, 2004), no training data was provided and systems must tag all the content words (noun, verb, adjective, and adverb) in running English texts with their correct WordNet senses. In SENSEVAL-2, the best performing system (Mihalcea and Moldovan, 2001) in the English all-words task achieved an accuracy of 69.0%, while in SENSEVAL-3, the best performing system (Decadt et al., 2004) achieved an accuracy of 65.2%. In SemEval-2007, which was the most recent SENSEVAL evaluation, a similar English all-words task was held, where systems had to provide the correct WordNet sense tag for all the verbs and head words of their arguments in running English texts. For this task, the best performing system (Tratz et al., 2007) achieved an accuracy of 59.1%. Results of these evaluations showed that state-of-the-art English all-words WSD systems performed with an accuracy of 60%–70%, using the fine-grained sense inventory of WordNet.

The low level of performance by these state-of-the-art WSD systems is a cause for concern, since WSD is supposed to be an enabling technology to be incorporated as a module into applications such as MT and IR. As mentioned earlier, one of the major reasons for the low performance is that these evaluation exercises adopted WordNet as the reference sense inventory, which is often too fine-grained. As an indication of this, inter-annotator agreement (ITA) reported for manual sense-tagging on these SENSEVAL English all-words datasets is typically in the mid-70s. To address this issue, a coarse-grained English all-words task (Navigli et al., 2007) was conducted during SemEval-2007. This task used a coarse-grained version of WordNet and reported an ITA of around 90%. We note that the best performing system (Chan et al., 2007b) of this task achieved a relatively high accuracy of 82.5%, highlighting the importance of having an appropriate level of sense granularity.

Another issue faced by current WSD systems is the lack of training data. We note that the top performing systems mentioned in the previous paragraphs are all based on supervised learning. With this approach, however, one would need to obtain a corpus where each ambiguous word occurrence is manually annotated with the correct sense, to serve as training data. Since it is time consuming to perform sense annotation of word occurrences, only a handful of sense-tagged corpora are publicly available. Among the existing sense-tagged corpora, the SEMCOR corpus (Miller et al., 1994) is one of the most widely used. In SEMCOR, content words have been manually tagged with WordNet senses. Current supervised WSD systems (which include all the top-performing systems in the English all-words task) usually rely on this relatively small manually annotated corpus for training examples, and this has inevitably affected the accuracy and scalability of current WSD systems.

Related to the problem of a lack of training data for WSD, there is also a lack of *test* data. Having a large amount of test data for evaluation is important to ensure the robustness and scalability of WSD systems. Due to the expensive process of manual sense-tagging, the SENSEVAL English all-words task evaluations were conducted on relatively small sets of evaluation data. For instance, the evaluation data of SENSEVAL-2 and SENSEVAL-3 English all-words task consists of 2,473 and 2,041 test examples respectively. In SemEval-2007, the fine-grained English all-words task consists of only 465 test examples, while the SemEval-2007 coarse-grained English all-words task consists of 2,269 test examples.

Hence, it is necessary to address the issues of sense granularity, and the lack of both training and test data. To this end, a recent large-scale annotation effort called the OntoNotes project (Hovy et al., 2006) was started. Building on the annotations from the Wall Street Journal (WSJ) portion of the Penn Treebank (Marcus et al., 1993), the project added several new layers of semantic annotations, such as coreference information, word senses, etc. In its first release (LDC2007T21) through the Linguistic Data Consortium (LDC), the project manually sense-tagged more than 40,000 examples belonging to hundreds of noun and verb types with an ITA of 90%, based on a coarse-grained sense inventory, where each word has an average of only 3.2 senses. Thus, besides providing WSD examples that were sense-tagged with a high ITA, the project also addressed the previously discussed issues of a lack of training and test data.

In this paper, we use the sense-tagged data provided by the OntoNotes project to investigate the accuracy achievable by current WSD systems when adopting a coarse-grained sense inventory. Through our experiments, we then highlight that domain adaptation for WSD is an important issue as it substantially affects the performance of a state-of-the-art WSD system which is trained on SEMCOR but evaluated on sense-tagged examples in OntoNotes. To address this issue, we then show that by combining a domain adaptation technique using feature augmentation with active learning, one only needs to annotate a small amount of in-domain examples to obtain a substantial improvement in the accuracy of the WSD system which is previously trained on out-of-domain examples.

The contributions of this paper are as follows. To our knowledge, this is the first large-scale WSD evaluation conducted that involves hundreds of word types and tens of thousands of sense-tagged examples, and that is based on a coarse-grained sense inventory. The present study also highlights the practical significance of domain adaptation in word sense disambiguation in the context of a large-scale empirical evaluation, and proposes an effective method to address the domain adaptation problem.

In the next section, we give a brief description of our WSD system. In Section 3, we describe experiments where we conduct both training and evaluation using data from OntoNotes. In Section 4, we investigate the WSD performance when we train our system on examples that are gathered from a different domain as compared to the OntoNotes evaluation data. In Section 5, we perform domain adaptation experiments using a recently introduced feature augmentation technique. In Section 6, we investigate the use of active learning to reduce the annotation effort required to adapt our WSD system to the domain of the OntoNotes data, before concluding in Section 7.

2 The WSD System

For the experiments reported in this paper, we follow the supervised learning approach of (Lee and Ng, 2002), by training an individual classifier for each word using the knowledge sources of local collocations, parts-of-speech (POS), and surrounding words.

For local collocations, we use 11 features: $C_{-1,-1}$, $C_{1,1}$, $C_{-2,-2}$, $C_{2,2}$, $C_{-2,-1}$, $C_{-1,1}$, $C_{1,2}$, $C_{-3,-1}$, $C_{-2,1}$, $C_{-1,2}$, and $C_{1,3}$, where $C_{i,j}$ refers to the ordered sequence of tokens in the local context of an ambiguous word *w*. Offsets *i* and *j* denote the starting and ending position (relative to *w*) of the sequence, where a negative (positive) offset refers to a token to its left (right). For parts-of-speech, we use 7 features: P_{-3}, P_{-2}, P_{-1}, P_0, P_1, P_2, P_3, where P_0 is the POS of *w*, and P_{-i} (P_i) is the POS of the *i*th token to the left (right) of *w*. For surrounding words, we consider all unigrams (single words) in the surrounding context of *w*. These words can be in a different sentence from *w*. For our experiments reported in this paper, we use support vector machines (SVM) as our learning algorithm, which was shown to achieve good WSD performance in (Lee and Ng, 2002; Chan et al., 2007b).

3 Training and Evaluating on OntoNotes

The annotated data of OntoNotes is drawn from the Wall Street Journal (WSJ) portion of the Penn Treebank corpus, divided into sections 00-24. These WSJ documents have been widely used in various NLP tasks such as syntactic parsing (Collins, 1999) and semantic role labeling (SRL) (Carreras and Marquez, 2005). In these tasks, the practice is to use documents from WSJ sections 02-21 as training data and WSJ section 23 as test data. Hence for our experiments reported in this paper, we follow this convention and use the annotated instances from WSJ sections 02-21 as our training data, and instances in WSJ section 23 as our test data.

Section	No. of word types	No. of word tokens	
		Individual	Cumulative
02	248	425	425
03	79	107	532
04	186	389	921
05	287	625	1546
06	224	446	1992
07	270	549	2541
08	177	301	2842
09	308	677	3519
10	648	3048	6567
11	724	4071	10638
12	740	4296	14934
13	749	4577	19511
14	710	3900	23411
15	748	4768	28179
16	306	576	28755
17	219	398	29153
18	266	566	29719
19	219	389	30108
20	288	536	30644
21	262	470	31114
23	685	3755	-

Table 1: Size of the sense-tagged data in the various WSJ sections.

As mentioned in Section 1, the OntoNotes data provided WSD examples for a large number of nouns and verbs, which are sense-tagged according to a coarse-grained sense inventory. In Table 1, we show the amount of sense-tagged data available from OntoNotes, across the various WSJ sections.[1] In the table, for each WSJ section, we list the number of word types, the number of sense-tagged examples, and the cumulative count on the number of

[1]We removed erroneous examples which were simply tagged with 'XXX' as sense-tag, or tagged with senses that were not found in the sense-inventory provided. Also, since we will be comparing against training on SEMCOR later (which was tagged using WordNet senses), we removed examples tagged with OntoNotes senses which were not mapped to WordNet senses. On the whole, about 7% of the original OntoNotes examples were removed as a result.

sense-tagged examples. From the table, we see that sections 02-21, which will be used as training data in our experiments, contain a total of slightly over 31,000 sense-tagged examples.

Using examples from sections 02-21 as training data, we trained our WSD system and evaluated on the examples from section 23. In our experiments, if a word type in section 23 has no training examples from sections 02-21, we randomly select an OntoNotes sense as the answer. Using these experimental settings, our WSD system achieved an accuracy of 89.1%. We note that this accuracy is much higher than the 60%–70% accuracies achieved by state-of-the-art English all-words WSD systems which are trained using the fine-grained sense inventory of WordNet. Hence, this highlights the importance of having an appropriate level of sense granularity.

Besides training on the entire set of examples from sections 02-21, we also investigated the performance achievable from training on various subsections of the data and show these results as "ON" in Figure 1. From the figure, we see that WSD accuracy increases as we add more training examples.

The fact that current state-of-the-art WSD systems are able to achieve a high level of performance is important, as this means that WSD systems will potentially be more usable for inclusion in end-applications. For instance, the high level of performance by syntactic parsers allows it to be used as an enabling technology in various NLP tasks. Here, we note that the 89.1% WSD accuracy we obtained is comparable to state-of-the-art syntactic parsing accuracies, such as the 91.0% performance by the statistical parser of Charniak and Johnson (2005).

4 Building WSD Systems with Out-of-Domain Data

Although our WSD system had achieved a high accuracy of 89.1%, this was achieved by training on a large amount (about 31,000) of manually sense annotated examples from sections 02-21 of the OntoNotes data. Further, all these training data and test data are gathered from the same domain of WSJ. In reality, however, since manual sense annotation is time consuming, it is not feasible to collect such a large amount of manually sense-tagged data for every domain of interest. Hence, in this section, we investigate the performance of our WSD system when it is trained on out-of-domain data.

In the English all-words task of the previous SENSEVAL evaluations (SENSEVAL-2, SENSEVAL-3, SemEval-2007), the best performing English all-words task systems with the highest WSD accuracy were trained on SEMCOR (Mihalcea and Moldovan, 2001; Decadt et al., 2004; Chan et al., 2007b). Hence, we similarly trained our WSD system on SEMCOR and evaluated on section 23 of the OntoNotes corpus. For those word types in section 23 which do not have training examples from SEMCOR, we randomly chose an OntoNotes sense as the answer. In training on SEMCOR, we have also ensured that there is a domain difference between our training and test data. This is because while the OntoNotes data was gathered from WSJ, which contains mainly business related news, the SEMCOR corpus is the sense-tagged portion of the Brown Corpus (BC), which is a mixture of several genres such as scientific texts, fictions, etc.

Evaluating on the section 23 test data, our WSD system achieved only 76.2% accuracy. Compared to the 89.1% accuracy achievable when we had trained on examples from sections 02-21, this is a substantially lower and disappointing drop of performance and motivates the need for domain adaptation.

The need for domain adaptation is a general and important issue for many NLP tasks (Daume III and Marcu, 2006). For instance, SRL systems are usually trained and evaluated on data drawn from the WSJ. In the CoNLL-2005 shared task on SRL (Carreras and Marquez, 2005), however, a task of training and evaluating systems on different domains was included. For that task, systems that were trained on the PropBank corpus (Palmer et al., 2005) (which was gathered from the WSJ), suffered a 10% drop in accuracy when evaluated on test data drawn from BC, as compared to the performance achievable when evaluated on data drawn from WSJ. More recently, CoNLL-2007 included a shared task on dependency parsing (Nivre et al., 2007). In this task, systems that were trained on Penn Treebank (drawn from WSJ), but evaluated on data drawn from a different domain (such as chemical abstracts and parent-child dialogues) showed a similar drop in performance. For research involving training and eval-

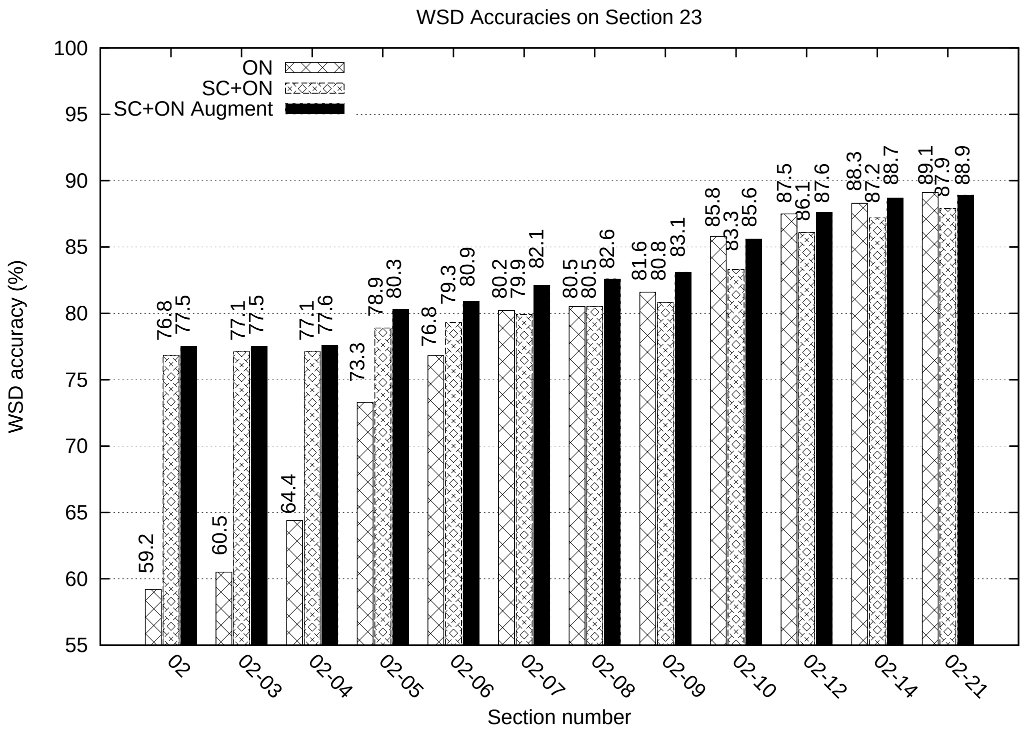


Figure 1: WSD accuracies evaluated on section 23, using SEMCOR and different OntoNotes sections as training data. ON: only OntoNotes as training data. SC+ON: SEMCOR and OntoNotes as training data, SC+ON Augment: Combining SEMCOR and OntoNotes via the Augment domain adaptation technique.

uating WSD systems on data drawn from different domains, several prior research efforts (Escudero et al., 2000; Martinez and Agirre, 2000) observed a similar drop in performance of about 10% when a WSD system that was trained on the BC part of the DSO corpus was evaluated on the WSJ part of the corpus, and vice versa.

In the rest of this paper, we perform domain adaptation experiments for WSD, focusing on domain adaptation methods that use in-domain annotated data. In particular, we use a feature augmentation technique recently introduced by Daume III (2007), and active learning (Lewis and Gale, 1994) to perform domain adaptation of WSD systems.

5 Combining In-Domain and Out-of-Domain Data for Training

In this section, we will first introduce the AUGMENT technique of Daume III (2007), before showing the performance of our WSD system with and without using this technique.

5.1 The AUGMENT technique for Domain Adaptation

The AUGMENT technique introduced by Daume III (2007) is a simple yet very effective approach to performing domain adaptation. This technique is applicable when one has access to training data from the source domain and a small amount of training data from the target domain.

The technique essentially augments the feature space of an instance. Assuming $\boldsymbol{x}$ is an instance and its original feature vector is $\Phi(\boldsymbol{x})$, the augmented feature vector for instance $\boldsymbol{x}$ is

$$\Phi'(\boldsymbol{x}) = \begin{cases} <\Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}), \mathbf{0}> & \text{if } \boldsymbol{x} \in D_s \\ <\Phi(\boldsymbol{x}), \mathbf{0}, \Phi(\boldsymbol{x})> & \text{if } \boldsymbol{x} \in D_t \end{cases},$$

where $\mathbf{0}$ is a zero vector of size $|\Phi(\boldsymbol{x})|$, D_s and D_t are the sets of instances from the source and target domains respectively. We see that the technique essentially treats the first part of the augmented feature space as holding general features that are not meant to be differentiated between different domains. Then, different parts of the augmented feature space are reserved for holding source domain specific, or target domain specific features. Despite its relative simplicity, this AUGMENT technique has been shown to outperform other domain adaptation techniques on various tasks such as named entity recognition, part-of-speech tagging, etc.

5.2 Experimental Results

As mentioned in Section 4, training our WSD system on SEMCOR examples gave a relatively low accuracy of 76.2%, as compared to the 89.1% accuracy obtained from training on the OntoNotes section 02-21 examples. Assuming we have access to some in-domain training data, then a simple method to potentially obtain better accuracies is to train on both the out-of-domain and in-domain examples. To investigate this, we combined the SEMCOR examples with various amounts of OntoNotes examples to train our WSD system and show the resulting "SC+ON" accuracies obtained in Figure 1. We also performed another set of experiments, where instead of simply combining the SEMCOR and OntoNotes examples, we applied the AUGMENT technique when combining these examples, treating SEMCOR examples as out-of-domain (source domain) data and OntoNotes examples as in-domain (target domain) data. We similarly show the resulting accuracies as "SC+ON Augment" in Figure 1.

Comparing the "SC+ON" and "SC+ON Augment" accuracies in Figure 1, we see that the AUGMENT technique *always* helps to improve the accuracy of our WSD system. Further, notice from the first few sets of results in the figure that when we have access to limited in-domain training examples from OntoNotes, incorporating additional out-of-domain training data from SEMCOR (either using the strategies "SC+ON" or "SC+ON Augment") achieves better accuracies than "ON". Significance tests using one-tailed paired t-test reveal that these accuracy improvements are statistically significant at the level of significance 0.01 (all significance tests in the rest of this paper use the same level of significance 0.01). These results validate the contribution of the SemCor examples. This trend continues till the result for sections 02-06.

The right half of Figure 1 shows the accuracy trend of the various strategies, in the unlikely event that we have access to a large amount of in-domain training examples. Although we observe that in this scenario, "ON" performs better than "SC+ON", "SC+ON Augment" continues to perform better than "ON" (where the improvement is statistically significant) till the result for sections 02-09. Beyond that, as we add more OntoNotes examples, significance testing reveals that the "SC+ON Augment" and "ON" strategies give comparable performance. This means that the "SC+ON Augment" strategy, besides giving good performance when one has few in-domain examples, does continue to perform well even when one has a large number of in-domain examples.

```
$D_S \leftarrow$ the set of SEMCOR training examples
$D_A \leftarrow$ the set of OntoNotes sections 02-21 examples
$D_T \leftarrow$ empty
while $D_A \neq \phi$
    $p_{min} \leftarrow \infty$
    $\Gamma \leftarrow$ WSD system trained on $D_S$ and $D_T$ using AUGMENT technique
    for each $d \in D_A$ do
        $\widehat{s} \leftarrow$ word sense prediction for $d$ using $\Gamma$
        $p \leftarrow$ confidence of prediction $\widehat{s}$
        if $p < p_{min}$ then
            $p_{min} \leftarrow p$, $d_{min} \leftarrow d$
        end
    end
    $D_A \leftarrow D_A - \{d_{min}\}$
    provide correct sense $s$ for $d_{min}$ and add $d_{min}$ to $D_T$
end
```

Figure 2: The active learning algorithm.

6 Active Learning with AUGMENT Technique

So far in this paper, we have seen that when we have access to some in-domain examples, a good strategy is to combine the out-of-domain and in-domain examples via the AUGMENT technique. This suggests that when one wishes to apply a WSD system to a new domain of interest, it is worth the effort to annotate a small number of examples gathered from the new domain. However, instead of randomly selecting in-domain examples to annotate, we could use active learning (Lewis and Gale, 1994) to help select in-domain examples to annotate. By doing so, we could minimize the manual annotation effort needed.

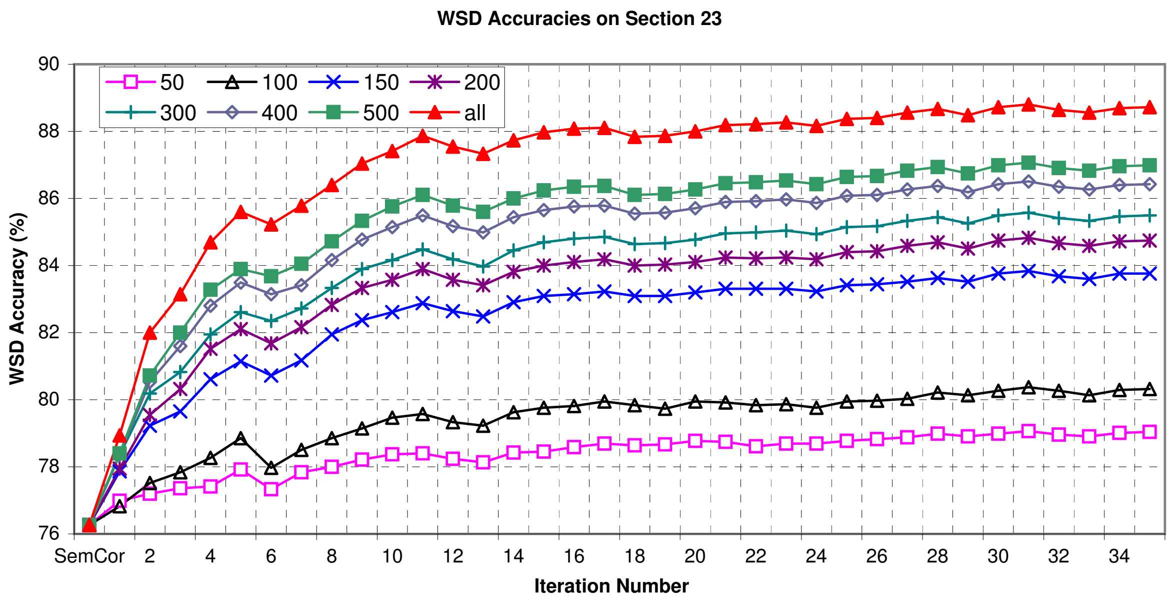


Figure 3: Results of applying active learning with the AUGMENT technique on different number of word types. Each curve represents the adaptation process of applying active learning on a certain number of most frequently occurring word types.

In WSD, several prior research efforts have successfully used active learning to reduce the annotation effort required (Zhu and Hovy, 2007; Chan and Ng, 2007; Chen et al., 2006; Fujii et al., 1998). With the exception of (Chan and Ng, 2007) which tried to adapt a WSD system trained on the BC part of the DSO corpus to the WSJ part of the DSO corpus, the other researchers simply applied active learning to reduce the annotation effort required and did not deal with the issue of adapting a WSD system to a new domain. Also, these prior research efforts only experimented with a few word types. In contrast, we perform active learning experiments on the hundreds of word types in the OntoNotes data, with the aim of adapting our WSD system trained on SEMCOR to the WSJ domain represented by the OntoNotes data.

For our active learning experiments, we use the *uncertainty sampling* strategy (Lewis and Gale, 1994), as shown in Figure 2. For our experiments, the SEMCOR examples will be our initial set of training examples, while the OntoNotes examples from sections 02-21 will be used as our pool of adaptation examples, from which we will select examples to annotate via active learning. Also, since we have found that the AUGMENT technique is useful in increasing WSD accuracy, we will apply the AUGMENT technique during each iteration of active learning to combine the SEMCOR examples and the selected adaptation examples.

As shown in Figure 2, we train an initial WSD system using only the set D_S of SEMCOR examples. We then apply our WSD system on the set D_A of OntoNotes adaptation examples. The example in D_A which is predicted with the lowest confidence will be removed from D_A and added to the set D_T of in-domain examples that have been selected via active learning thus far. We then use the AUGMENT technique to combine the set of examples in D_S and D_T to train a new WSD system, which is then applied again on the set D_A of remaining adaptation examples, and this active learning process continues until we have used up all the adaptation examples. Note that because we are using OntoNotes sections 02-21 (which have already been sense-tagged beforehand) as our adaptation data, the annotation of the selected example during each active learning iteration is simply simulated by referring to its tagged sense.

6.1 Experimental Results

As mentioned earlier, we use the examples in OntoNotes sections 02-21 as our adaptation examples during active learning. Hence, we perform active learning experiments on *all* the word types that have sense-tagged examples from OntoNotes sections 02-21, and show the evaluation results on OntoNotes section 23 as the topmost "all" curve in Figure 3. Since our aim is to reduce the human annotation effort required in adapting a WSD system to a new domain, we may not want to perform active learning on all the word types in practice. Instead, we can maximize the benefits by performing active learning only on the more frequently occurring word types. Hence, in Figure 3, we also show via various curves the results of applying active learning only to various sets of word types, according to their frequency, or number of sense-tagged examples in OntoNotes sections 02-21. Note that the various accuracy curves in Figure 3 are plotted in terms of evaluation accuracies over all the test examples in OntoNotes section 23, hence they are directly comparable to the results reported thus far in this paper. Also, since the accuracies for the various curves stabilize after 35 active learning iterations, we only show the results of the first 35 iterations.

From Figure 3, we note that by performing active learning on the set of 150 most frequently occurring word types, we are able to achieve a WSD accuracy of 82.6% after 10 active learning iterations. Note that in Section 4, we mentioned that training only on the out-of-domain SEMCOR examples gave an accuracy of 76.2%. Hence, we have gained an accuracy improvement of 6.4% (82.6% − 76.2%) by just using 1,500 in-domain OntoNotes examples. Compared with the 12.9% (89.1% − 76.2%) improvement in accuracy achieved by using all 31,114 OntoNotes sections 02-21 examples, we have obtained half of this maximum increase in accuracy, by requiring only about 5% (1,500/31,114) of the total number of sense-tagged examples. Based on these results, we propose that when there is a need to apply a previously trained WSD system to a different domain, one can apply the AUGMENT technique with active learning on the most frequent word types, to greatly reduce the annotation effort required while obtaining a substantial improvement in accuracy.

7 Conclusion

Using the WSD examples made available through OntoNotes, which are sense-tagged according to a coarse-grained sense inventory, we show that our WSD system is able to achieve a high accuracy of 89.1% when we train and evaluate on these examples. However, when we apply a WSD system that is trained on SEMCOR, we suffer a substantial drop in accuracy, highlighting the need to perform domain adaptation. We show that by combining the AUGMENT domain adaptation technique with active learning, we are able to effectively reduce the amount of annotation effort required for domain adaptation.

References

M. Carpuat and D. Wu. 2007. Improving Statistical Machine Translation Using Word Sense Disambiguation. In *Proc. of EMNLP-CoNLL07*, pages 61–72.

X. Carreras and L. Marquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proc. of CoNLL-2005*, pages 152–164.

Y. S. Chan and H. T. Ng. 2007. Domain Adaptation with Active Learning for Word Sense Disambiguation. In *Proc. of ACL07*, pages 49–56.

Y. S. Chan, H. T. Ng, and D. Chiang. 2007a. Word Sense Disambiguation Improves Statistical Machine Translation. In *Proc. of ACL07*, pages 33–40.

Y. S. Chan, H. T. Ng, and Z. Zhong. 2007b. NUS-PT: Exploiting Parallel Texts for Word Sense Disambiguation in the English All-Words Tasks. In *Proc. of SemEval-2007*, pages 253–256.

E. Charniak and M. Johnson. 2005. Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking. In *Proc. of ACL05*, pages 173–180.

J. Y. Chen, A. Schein, L. Ungar, and M. Palmer. 2006. An Empirical Study of the Behavior of Active Learning for Word Sense Disambiguation. In *Proc. of HLT/NAACL06*, pages 120–127.

M. Collins. 1999. *Head-Driven Statistical Model for Natural Language Parsing*. PhD dissertation, University of Pennsylvania.

H. Daume III and D. Marcu. 2006. Domain Adaptation for Statistical Classifiers. *Journal of Artificial Intelligence Research*, 26:101–126.

H. Daume III. 2007. Frustratingly Easy Domain Adaptation. In *Proc. of ACL07*, pages 256–263.

B. Decadt, V. Hoste, and W. Daelemans. 2004. GAMBL, Genetic Algorithm Optimization of Memory-Based WSD. In *Proc. of SENSEVAL-3*, pages 108–112.

G. Escudero, L. Marquez, and G. Riagu. 2000. An Empirical Study of the Domain Dependence of Supervised Word Sense Disambiguation Systems. In *Proc. of EMNLP/VLC00*, pages 172–180.

A. Fujii, K. Inui, T. Tokunaga, and H. Tanaka. 1998. Selective Sampling for Example-based Word Sense Disambiguation. *Computational Linguistics*, 24(4).

E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel. 2006. OntoNotes: The 90% solution. In *Proc. of HLT-NAACL06*, pages 57–60.

Y. K. Lee and H. T. Ng. 2002. An Empirical Evaluation of Knowledge Sources and Learning Algorithms for Word Sense Disambiguation. In *Proc. of EMNLP02*, pages 41–48.

D. D. Lewis and W. A. Gale. 1994. A Sequential Algorithm for Training Text Classifiers. In *Proc. of SIGIR94*.

M. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

D. Martinez and E. Agirre. 2000. One Sense per Collocation and Genre/Topic Variations. In *Proc. of EMNLP/VLC00*, pages 207–215.

R. Mihalcea and D. Moldovan. 2001. Pattern Learning and Active Feature Selection for Word Sense Disambiguation. In *Proc. of SENSEVAL-2*, pages 127–130.

G. A. Miller, M. Chodorow, S. Landes, C. Leacock, and R. G. Thomas. 1994. Using a Semantic Concordance for Sense Identification. In *Proc. of ARPA Human Language Technology Workshop*, pages 240–243.

G. A. Miller. 1990. WordNet: An On-line Lexical Database. *International Journal of Lexicography*, 3(4):235–312.

R. Navigli, K. C. Litkowski, and O. Hargraves. 2007. SemEval-2007 Task 07: Coarse-Grained English All-Words Task. In *Proc. of SemEval-2007*, pages 30–35.

J. Nivre, J. Hall, S. Kubler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 Shared Task on Dependency Parsing. In *Proc. of EMNLP-CoNLL07*, pages 915–932.

M. Palmer, C. Fellbaum, S. Cotton, L. Delfs, and H. T. Dang. 2001. English Tasks: All-Words and Verb Lexical Sample. In *Proc. of SENSEVAL-2*, pages 21–24.

M. Palmer, D. Gildea, and P. Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–105.

B. Snyder and M. Palmer. 2004. The English All-Words Task. In *Proc. of SENSEVAL-3*, pages 41–43.

S. Tratz, A. Sanfilippo, M. Gregory, A. Chappell, C. Posse, and P. Whitney. 2007. PNNL: A Supervised Maximum Entropy Approach to Word Sense Disambiguation. In *Proc. of SemEval-2007*, pages 264–267.

J. B. Zhu and E. Hovy. 2007. Active Learning for Word Sense Disambiguation with Methods for Addressing the Class Imbalance Problem. In *Proc. of EMNLP-CoNLL07*, pages 783–790.

Graph-based Analysis of Semantic Drift in Espresso-like Bootstrapping Algorithms

Mamoru Komachi
NAIST, Japan
mamoru-k@is.naist.jp

Taku Kudo
Google Inc.
taku@google.com

Masashi Shimbo
NAIST, Japan
shimbo@is.naist.jp

Yuji Matsumoto
NAIST, Japan
matsu@is.naist.jp

Abstract

Bootstrapping has a tendency, called *semantic drift*, to select instances unrelated to the seed instances as the iteration proceeds. We demonstrate the semantic drift of bootstrapping has the same root as the topic drift of Kleinberg's HITS, using a simplified graph-based reformulation of bootstrapping. We confirm that two graph-based algorithms, the von Neumann kernels and the regularized Laplacian, can reduce semantic drift in the task of word sense disambiguation (WSD) on Senseval-3 English Lexical Sample Task. Proposed algorithms achieve superior performance to *Espresso* and previous graph-based WSD methods, even though the proposed algorithms have less parameters and are easy to calibrate.

1 Introduction

In recent years machine learning techniques become widely used in natural language processing (NLP). These techniques offer various ways to exploit large corpora and are known to perform well in many tasks. However, these techniques often require tagged corpora, which are not readily available to many languages. So far, reducing the cost of human annotation is one of the important problems for building NLP systems.

To mitigate the problem of hand-tagging resources, semi(or minimally)-supervised and unsupervised techniques have been actively studied. Hearst (1992) first presented a bootstrapping method which requires only a small amount of instances (seed instances) to start with, but can easily multiply the number of tagged instances with minimal human annotation cost, by iteratively applying the following phases: pattern induction, pattern ranking/selection, and instance extraction. Bootstrapping has been widely adopted in NLP applications such as word sense disambiguation (Yarowsky, 1995), named entity recognition (Collins and Singer, 1999) and relation extraction (Riloff and Jones, 1999; Pantel and Pennacchiotti, 2006).

However, it is known that bootstrapping often acquires instances not related to seed instances. For example, consider the task of collecting the names of common tourist sites from web corpora. Given words like "Geneva" and "Bali" as seed instances, bootstrapping would eventually learn *generic patterns* such as "pictures" and "photos," which also co-occur with many other unrelated instances. The subsequent iterations would likely acquire frequent words that co-occur with these generic patterns, such as "Britney Spears." This phenomenon is called *semantic drift* (Curran et al., 2007).

A straightforward approach to avoid semantic drift is to terminate iterations before hitting generic patterns, but the optimal number of iterations is task dependent and is hard to come by. The recently proposed *Espresso* (Pantel and Pennacchiotti, 2006) algorithm incorporates sophisticated scoring functions to cope with generic patterns, but as Komachi and Suzuki (2008) pointed out, Espresso still shows semantic drift unless iterations are terminated appropriately.

Another deficiency in bootstrapping is its sensitivity to many parameters such as the number of seed instances, the stopping criterion of iteration, the number of instances and patterns selected on each iteration, and so forth. These parameters also need to be calibrated for each task.

In this paper, we present a graph-theoretic analysis of Espresso-like bootstrapping algorithms. We argue that semantic drift is inherent in these algorithms, and propose to use two graph-based algorithms that are theoretically less prone to semantic drift, as an alternative to bootstrapping.

After a brief review of related work in Section 2, we analyze in Section 3 a bootstrapping algorithm (*Simplified Espresso*) which can be thought of as a degenerate version of Espresso. Simplified Espresso is simple enough to allow an algebraic treatment, and its equivalence to Kleinberg's HITS algorithm (Kleinberg, 1999) is shown. An implication of this equivalence is that semantic drift in this bootstrapping algorithm is essentially the same phenomenon as topic drift observed in link analysis. Another implication is that semantic drift is inevitable in Simplified Espresso as it converges to the same score vector regardless of seed instances.

The original Espresso also suffers from the same problem as its simplified version does. It incorporates heuristics not present in Simplified Espresso to reduce semantic drift, but these heuristics have limited effect as we demonstrate in Section 3.3.

In Section 4, we propose two graph-based algorithms to reduce semantic drift. These algorithms are used in link analysis community to reduce the effect of topic drift. In Section 5 we apply them to the task of word sense disambiguation on Senseval-3 Lexical Sample Task and verify that they indeed reduce semantic drift. Finally, we conclude our work in Section 6.

2 Related Work

2.1 Overview of Bootstrapping

Bootstrapping (or self-training) is a general framework for reducing the requirement of manual annotation. Hearst (1992) described a bootstrapping procedure for extracting words in hyponym (*is-a*) relation, starting with three manually given lexico-syntactic patterns.

The idea of learning with a bootstrapping method was adopted for many tasks. Yarowsky (1995) presented an unsupervised WSD system which rivals supervised techniques. Abney (2004) presented a thorough discussion on the Yarowsky algorithm. He extended the original Yarowsky algorithm to a new family of bootstrapping algorithms that are mathematically well understood.

Li and Li (2004) proposed a method called *Bilingual Bootstrapping*. It makes use of a translation dictionary and a comparable corpus to help disambiguate word senses in the source language, by exploiting the asymmetric many-to-many sense mapping relationship between words in two languages.

Curran et al. (2007) presented an algorithm called *Mutual Exclusion Bootstrapping*, which minimizes semantic drift using mutual exclusion between semantic classes of learned instances. They prepared a list of so-called *stop classes* similar to a stop word list used in information retrieval to help bound the semantic classes. Stop classes are sets of terms known to cause semantic drift in particular semantic classes. However, stop classes vary from task to task and domain to domain, and human intervention is essential to create an effective list of stop classes.

A major drawback of bootstrapping is the lack of principled method for selecting optimal parameter values (Ng and Cardie, 2003; Banko and Brill, 2001). Also, there is an issue of generic patterns which deteriorates the quality of acquired instances. Previously proposed bootstrapping algorithms differ in how they deal with the problem of semantic drift. We will take recently proposed Espresso algorithm as the example to explain common configuration for bootstrapping in detail.

2.2 The Espresso Algorithm

Pantel and Pennachiotti (2006) proposed a bootstrapping algorithm called Espresso to learn binary semantic relations such as *is-a* and *part-of* from a corpus. What distinguishes Espresso from other bootstrapping algorithms is that it benefits from generic patterns by using a principled measure of instance and pattern reliability. The key idea of Espresso is recursive definition of pattern-instance scoring metrics. The reliability scores of pattern p and instance i, denoted respectively as $r_\pi(p)$ and

$r_\iota(i)$, are given as follows:

$$r_\pi(p) = \frac{\sum_{i \in I} \frac{pmi(i,p)}{\max\ pmi} r_\iota(i)}{|I|} \quad (1)$$

$$r_\iota(i) = \frac{\sum_{p \in P} \frac{pmi(i,p)}{\max\ pmi} r_\pi(p)}{|P|} \quad (2)$$

where

$$pmi(i,p) = \log_2 \frac{|i,p|}{|i,*||*,p|} \quad (3)$$

is pointwise mutual information between i and p, P and I are sets of patterns and instances, and $|P|$ and $|I|$ are the numbers of patterns and instances, respectively. $|i,*|$ and $|*,p|$ are the frequencies of pattern p and instance i in a given corpus, respectively, and $|i,p|$ is the frequency of pattern p which co-occurs with instance i. $\max\ pmi$ is a maximum value of the pointwise mutual information over all instances and patterns. The intuition behind these definitions is that a reliable pattern co-occurs with many reliable instances, and a reliable instance co-occurs with many reliable patterns.

Espresso and other bootstrapping methods iterate the following three phases: *pattern induction, pattern ranking/selection, and instance extraction.*

We describe these phases below, along with the parameters that controls each phase.

Phase 1. Pattern Induction Induce patterns from a corpus given seed instances. Patterns may be surface text patterns, lexico-syntactic patterns, and/or just features.

Phase 2. Pattern Ranking/Selection Create a pattern ranker from a corpus using instances as features and select patterns which co-occur with seed instances for the next instance extraction phase. The main issue here is to avoid ranking generic patterns high and to choose patterns with high relatedness to the seed instances. Parameters and configurations: (a) *a pattern scoring metrics* and (b) *the number of patterns to use for extraction of instances*.

Phase 3. Instance Extraction Select high-confidence instances to the seed instance set. It is desirable to keep only high-confidence instances at this phase, as they are used as seed instances for the next iteration. Optionally, instances can be cumulatively obtained on each iteration to retain highly relevant instances learned in early iterations. Parameters and configurations: (c) *instance scoring metrics*, (d) *whether to retain extracted instances on each iteration or not*, and (e) *the number of instances to pass to the next iteration*.

```
input:
    seed vector $\mathbf{i}_0$
    pattern-instance co-occurrence matrix $M$
output:
    instance and pattern score vectors $\mathbf{i}$ and $\mathbf{p}$

1:  $\mathbf{i} = \mathbf{i}_0$
2:  loop
3:      $\mathbf{p} \leftarrow M\mathbf{i}$
4:      Normalize $\mathbf{p}$
5:      $\mathbf{i} \leftarrow M^T\mathbf{p}$
6:      Normalize $\mathbf{i}$
7:      if $\mathbf{i}$ and $\mathbf{p}$ have both converged then
8:          return $\mathbf{i}$ and $\mathbf{p}$
9:      end if
10: end loop
```

Figure 1: A simple bootstrapping algorithm

Bootstrapping iterates the above three phases several times until stopping criteria are met. Acquired instances tend to become noisy as the iteration proceeds, so it is important to terminate before semantic drift occurs. Thus, we have another configuration: (f) *stopping criterion*.

Espresso uses Equations (1) for (a) and (2) for (c) respectively, whereas other parameters rely on the tasks and need calibration. Even though Espresso greatly improves recall while keeping high precision by using these pattern and instance scoring metrics, Komachi and Suzuki (2008) observed that extracted instances matched against generic patterns may become erroneous after tens of iterations, showing the difficulty of applying bootstrapping methods to different domains.

3 Analysis of an Espresso-like Bootstrapping Algorithm

3.1 Simplified Espresso

Let us consider a simple bootstrapping algorithm illustrated in Figure 1, in order to elucidate the cause of semantic drift.

As before, let $|I|$ and $|P|$ be the numbers of instances and patterns, respectively. The algorithm takes a seed vector $\mathbf{i}_0$, and a pattern-instance co-occurrence matrix M as input. $\mathbf{i}_0$ is a $|I|$-dimensional vector with 1 at the position of seed instances, and 0 elsewhere. M is a $|P| \times |I|$-matrix whose (p, i)-element $[M]_{pi}$ holds the (possibly reweighted) number of co-occurrence of pattern p and instance i in the corpus. If both $\mathbf{i}$ and $\mathbf{p}$ have converged, the algorithm returns the pair of $\mathbf{i}$ and $\mathbf{p}$ as output.

This algorithm, though simple, can encode Espresso's update formulae (1) and (2) as Steps 3 through 6 if we pose

$$[M]_{pi} = \frac{pmi(i, p)}{\max pmi}, \tag{4}$$

and normalize $\mathbf{p}$ and $\mathbf{i}$ in Steps 4 and 6 by

$$\mathbf{p} \leftarrow \mathbf{p}/|I| \quad \text{and} \quad \mathbf{i} \leftarrow \mathbf{i}/|P|, \tag{5}$$

respectively.

This specific instance of the algorithm of Figure 1, obtained by specialization through Equations (4) and (5), will be henceforth referred to as *Simplified Espresso*. Indeed, it is an instance of the original Espresso in which the iteration is not terminated until convergence, all instances are carried over to the next iteration, and instances are not cumulatively learned.

3.2 Simplified Espresso as Link Analysis

Let n denote the number of times Steps 2–10 are iterated. Plugging (4) and (5) into Steps 3–6, we see that the score vector of instances after the nth iteration is

$$\mathbf{i}_n = A^n \mathbf{i}_0 \tag{6}$$

where

$$A = \frac{1}{|I||P|} M^T M. \tag{7}$$

Suppose matrix A is irreducible; i.e., the graph induced by taking A as the adjacency matrix is connected. If n is increased and $\mathbf{i}_n$ is normalized on each iteration, $\mathbf{i}_n$ tends to the principal eigenvector of A. This implies that no matter what seed instances are input, the algorithm will end up with the same ranking of instances, if it is run until convergence. Because $A = \frac{M^T M}{|I||P|}$, the principal eigenvector of A is identical to the authority vector of HITS (Kleinberg, 1999) algorithm run on the graph induced by M. [1] This similarity of Equations (1), (2) and HITS is not discussed in (Pantel and Pennacchiotti, 2006).

As a consequence of the above discussion, semantic drift in simplified Espresso seems to be inevitable as the iteration proceeds, since the principal eigenvector of A need not resemble seed vector $\mathbf{i}_0$. A similar phenomenon is reported for HITS and is known as *topic drift*, in which pages of the dominant topic are ranked high regardless of the given query. (Bharat and Henzinger, 1998)

Unlike HITS and Simplified Espresso, however, Espresso and other bootstrapping algorithms (Yarowsky, 1995; Riloff and Jones, 1999), incorporate heuristics so that only patterns and instances with high confidence score are carried over to the next iteration.

3.3 Convergence Process of Espresso

To investigate the effect of semantic drift on Espresso with and without the heuristics of selecting the most confident instances on each iteration (i.e., the original Espresso and Simplified Espresso of Section 3.2), we apply them to the task of word sense disambiguation of word "bank" in the Senseval-3 Lexical Sample (S3LS) Task data.[2] There are 394 instances of word "bank" and their occurring context in this dataset, and each of them is annotated with its true sense. Of the ten senses of *bank,* the most frequent is the bank as in "bank of the river." We use the standard training-test split provided with the data set.

We henceforth denote Espresso with the following filtering strategy as *Filtered Espresso* to stress the distinction from Simplified Espresso. For Filtered Espresso, we cleared all but the 100 top-scoring instances in the instance vector on each iteration, and the number of non-zeroed instance scores

[1] As long as the relative magnitude of the components of vector $\mathbf{i}_n$ is preserved, the vector can be normalized in any way on each iteration. Hence HITS and Simplified Espresso use different normalization but both converge to the principal eigenvector of A.

[2] http://www.senseval.org/senseval3/data.html

grows by 100 on each iteration. On the other hand, we cleared all but the 20 top-scoring patterns in the pattern vector on each iteration, and the number of non-zeroed pattern scores grows by 1 on each iteration following (Pantel and Pennacchiotti, 2006).[3] The values of other parameters (b), (d), (e) and (f) remains the same as those for simplified Espresso in Section 3.1.

The task of WSD is to correctly predict the senses of test instances whose true sense is hidden from the system, using training data and their true senses. To predict the sense of a given instance i, we apply k-nearest neighbor algorithm.

Given a test instance i, its sense is predicted with the following procedure:

1. Compute the instance-pattern matrix M from the entire set of instances. We defer the details of this step to Section 5.2.
2. Run Simplified- and Filtered Espresso using the given instance i as the only seed instance.
3. After the termination of the algorithm, select k training instances with the highest scores in the score vector $\mathbf{i}$ output by the algorithm.
4. Since the selected k instances are training instances, their true senses are accessible. Choose the majority sense s from these k instances, and output s as the prediction for the given instance i. When there is a tie, output the sense of the instance with the highest score in $\mathbf{i}$. Note that only Step 4 uses sense information.

Figure 2 shows the convergence process of Simplified- and Filtered Espresso. X-axis indicates the number of bootstrapping iterations and Y-axis indicates the recall, which in this case equals precision, as the coverage is 100% in all cases.

[3]We conducted preliminary experiment to find these parameters to maximize the performance of Filtered Espresso. (These numbers are different from the original Espresso (Pantel and Pennacchiotti, 2006).) The number of initial patterns is relatively large because of a data sparseness problem in WSD, unlike relation extraction and named entity recognition. Also, WSD basically uses more features than relation extraction and thus it is hard to determine the stopping criterion based on the number and scores of patterns, as (Pantel and Pennacchiotti, 2006) does.

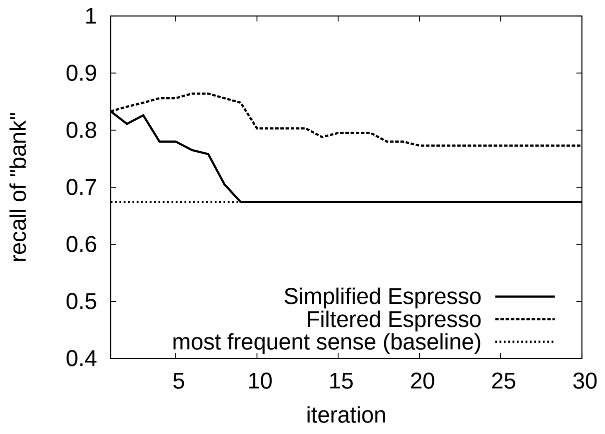


Figure 2: Recall of Simplified- and Filtered Espresso

Simplified Espresso tends to select the most frequent sense as the iteration proceeds, and after nine iterations it selects the most frequent sense ("the bank of the river") regardless of the seed instances. As expected from the discussion in Section 3.2, generic patterns gradually got more weight and semantic drift occurred in later iterations. Indeed, the ranking of the instances after convergence was identical to the HITS authority ranking computed from instance-pattern matrix M (i.e., the ranking induced by the dominant eigenvector of $M^T M$).

On the other hand, Filtered Espresso suffers less from semantic drift. The final recall achieved was 0.773 after convergence on the 20th iteration, outperforming the most-frequent sense baseline by 0.10. However, a closer look reveals that the filtering heuristics is limited in effectiveness.

Figure 3 plots the learning curve of Filtered Espresso on the set of test instances. We show recall ($\frac{|correct\ instances|}{|total\ true\ instances|}$) of each sense to see how Filtered Espresso tends to select the most frequent sense. If semantic drift takes place, the number of instances predicted as the most frequent sense should increase as the iteration proceeds, resulting in increased recall on the most frequent sense and decreased recall on other senses. Figure 3 exactly exhibit this trend, meaning that Filtered Espresso is not completely free from semantic drift. Figure 2 also shows that the recall of Filtered Espresso starts to decay after the seventh iteration.

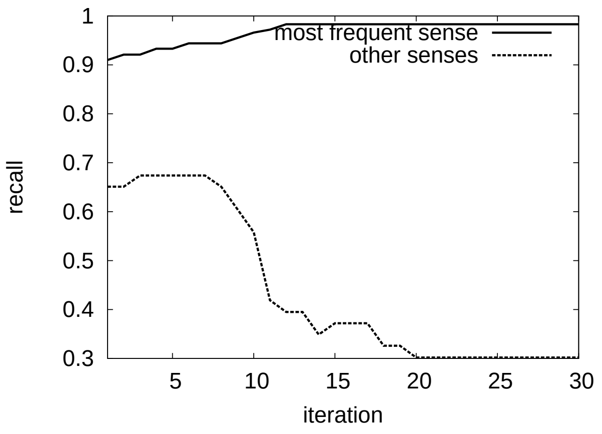


Figure 3: Recall of Filtered Espresso on the instances having "bank of the river" and other senses

4 Two Graph-based Algorithms for Exploiting Generic Patterns

We explore two graph-based methods which have the advantage of Espresso to harness the property of generic patterns by the mutual recursive definition of instance and pattern scores. They also have less parameters than bootstrapping, and are less prone to semantic drift.

4.1 Von Neumann Kernel

Kandola et al. (2002) proposed the *von Neumann kernels* for measuring similarity of documents using words. If we apply the von Neumann kernels to the pattern-instance co-occurrence matrix instead of the document-word matrix, the relative importance of an instance to seed instances can be estimated.

Let $A = M^T M$ be the instance similarity matrix obtained from pattern-instance matrix M, and λ be the principal eigenvalue of A. The *von Neumann kernel matrix* K_β with diffusion factor β $(0 \leq \beta < \lambda^{-1})$ is defined as follows:

$$K_\beta \;=\; A \sum_{n=0}^{\infty} \beta^n A^n = A(I - \beta A)^{-1}. \quad (8)$$

The similarity between two instances i, j is given by the (i, j) element of K_β. Hence, the i-th column vector can be used as the score vector for seed instance i.

Ito et al. (2005) showed that the von Neumann kernels represent a mixture of the co-citation relatedness and Kleinberg's HITS importance. They compute the weighted sum of all paths between two nodes in the co-citation graph induced by $A = M^T M$. The $(M^T M)^n$ term of smaller n corresponds to the relatedness to the seed instances, and the $(M^T M)^n$ term of larger n corresponds to HITS importance. The von Neumann kernels calculate the weighted sum of $(M^T M)^n$ from $n = 1$ to ∞, and therefore smaller diffusion factor β results in ranking by relatedness, and larger β returns ranking by HITS importance.

In NLP literature, Schütze (1998) introduced the notion of first- and second-order co-occurrence. First-order co-occurrence is a context which directly co-occurs with a word, whereas second-order co-occurrence is a context which occurs with the (contextual) words that co-occur with a word. Higher-order co-occurrence information is less sparse and more robust than lower-order co-occurrence, and thus is useful for a proximity measure.

Given these definitions, we see that the $(M^T M)^n$ term of smaller n corresponds to lower-order co-occurrence, which is accurate but sparse, and the $(M^T M)^n$ term of larger n corresponds to higher-order co-occurrence, which is dense but possibly giving too much weight on unrelated instances extracted by generic patterns.

As a result, it is expected that setting diffusion factor β to a small value prevents semantic drift and also takes higher order pattern vectors into account. We verify this claim in Section 5.3.

4.2 Regularized Laplacian Kernel

The von Neumann kernels can be regarded as a mixture of relatedness and importance, and diffusion factor β controls the trade-off between relatedness and importance. In practice, however, setting the right parameter value becomes an issue. We solve this problem by the regularized Laplacian (Smola and Kondor, 2003; Chebotarev and Shamis, 1998), which are stable across diffusion factors and can safely benefit from generic patterns.

Let G be a weighted undirected graph whose adjacency (weight) matrix is a symmetric matrix A. The (combinatorial) graph Laplacian L of a graph G is defined as follows:

$$L = D - A \quad (9)$$

where D is a diagonal matrix, and the ith diagonal

Table 1: Recall of predicted labels of *bank*

algorithm	MFS	others
Simplified Espresso	100.0	0.0
Filtered Espresso	100.0	30.2
Filtered Espresso (optimal stopping)	94.4	67.4
von Neumann kernels	92.1	65.1
regularized Laplacian	92.1	62.8

element $[D]_{ii}$ is given by

$$[D]_{ii} = \sum_j [A]_{ij}. \tag{10}$$

Here, $[A]_{ij}$ stands for the (i, j) element of A. By replacing A with $-L$ in Equation (8) and deleting the first A, we obtain a *regularized Laplacian kernel* [4].

$$R_\beta = \sum_{n=0}^{\infty} \beta^n (-L)^n = (I + \beta L)^{-1} \tag{11}$$

Again, $\beta(\geq 0)$ is called the diffusion factor.

Both the regularized Laplacian and the von Neumann kernels compute all the possible paths in a graph, and consequently they can calculate influence between nodes in a long distance in the graph. Also, Equations (9) and (10) show that the negative Laplacian $-L$ can be regarded as a modification to the graph G with the weight of self-loops re-weighted to negative values. In this modified graph, if an instance co-occurs with a pattern which also co-occurs with a large number of other instances, a self-loop of a node in the instance similarity graph induced by $M^T M$ will receive a higher negative weight. In other words, instances co-occurring with generic patterns will get less weight in the regularized Laplacian than in the von Neumann kernels.

5 Experiments and Results

5.1 Experiment 1: Reducing Semantic Drift

We test the von Neumann kernels and the regularized Laplacian on the same task as we used in Section 3.3; i.e., word sense disambiguation of word "bank." During the training phase, a pattern-instance matrix M was constructed using the training and testing data from Senseval-3 Lexical Sample (S3LS) Task. The (i, j) element of M of both kernels is set to pointwise mutual information of a pattern i and an instance j, just the same as in Espresso. Recall is used in evaluation.[5] The diffusion parameter β is set to 10^{-5} and 10^{-2} for the von Neumann kernels and the regularized Laplacian, respectively.

Table 1 illustrates how well the proposed methods reduce semantic drift, just the same as the experiment of Figure 3 in Section 3.3. We evaluate the recall on predicting the most frequent sense (MFS) and the recall on predicting other less frequent senses (others). For Filtered Espresso, two results are shown: the result on the seventh iteration, which maximizes the performance (Filtered Espresso (optimal stopping)), and the one after convergence. As in Section 3.3, if semantic drift occurs, recall of prediction on the most frequent sense increases while recall of prediction on other senses declines. Even Filtered Espresso was affected by semantic drift, which is again a consequence of the inherent graphical nature of Espresso-like bootstrapping algorithms. On the other hand, both proposed methods succeeded to balance the most frequent sense and other senses. Filtered Espresso at the optimal number of iterations achieved the best performance. Nevertheless, the number of iterations has to be estimated separately.

5.2 Experiment 2: WSD Benchmark Data

We conducted experiments on the task of word sense disambiguation of S3LS data, this time not just on the word "bank" but on all target nouns in the data, following (Agirre et al., 2006). We used two types of patterns.

Unordered single words (bag-of-words) We used all single words (unigrams) in the provided context from S3LS data sets. Each word in the context constructs one pattern. The pattern corresponding to a word w is set to 1 if it appears in the context of instance i. Words were lowercased and pre-processed with the Porter Stemmer[6].

[4]It has been reported that normalization of A improves performance in application (Johnson and Zhang, 2007), so we normalize L by $L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$.

[5]Again, recall equals precision in this case as the coverage is 100% in all cases.

[6]http://tartarus.org/~martin/PorterStemmer/def.txt

Table 2: Comparison of WSD algorithms

algorithm	Recall
most frequent sense	54.5
HyperLex (Véronis, 2004)	64.6
PageRank (Agirre et al., 2006)	64.5
Simplified Espresso	44.1
Filtered Espresso	46.9
Filtered Espresso (optimal stopping)	66.5
von Neumann kernels ($\beta = 10^{-5}$)	**67.2**
regularized Laplacian ($\beta = 10^{-2}$)	67.1

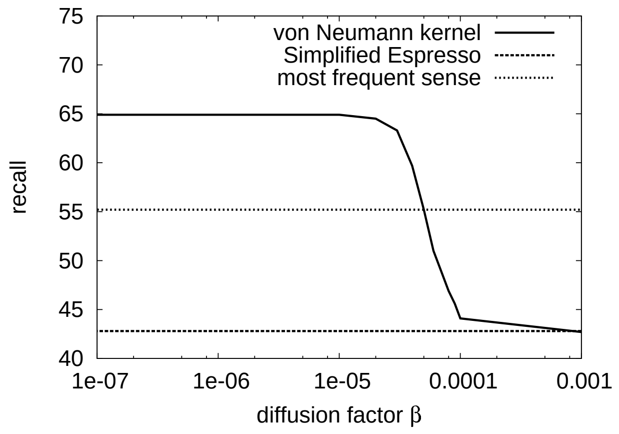


Figure 4: Recall of the von Neumann kernels with a different diffusion factor β on S3LS WSD task

Local collocations A local collocation refers to the ordered sequence of tokens in the local, narrow context of the target word. We allowed a pattern to have wildcard expressions like "sale of * *interest* in * *" for the target word *interest*. We set the window size to ± 3 by a preliminary experiment.

We report the results of Filtered Espresso both after convergence, and with its optimal number of iterations to show the upper bound of its performance.

Table 2 compares proposed methods with Espresso with various configurations. The proposed methods outperform by a large margin the most frequent sense baseline and both Simplified- and Filtered Espresso. This means that the proposed methods effectively prevent semantic drift.

Also, Filtered Espresso without early stopping shows more or less identical performance to Simplified Espresso. It is implied that the heuristics of filtering and early stopping is a crucial step not to select generic patterns in Espresso, and the result is consistent with the experiment of convergence process of Espresso in Section 3.3.

Filtered Espresso halted after the seventh iteration (Filtered Espresso (optimal stopping)) is comparable to the proposed methods. However, in bootstrapping, not only the number of iterations but also a large number of parameters must be adjusted for each task and domain. This shortcoming makes it hard to adapt bootstrapping in practical cases. One of the main advantages of the proposed methods is that they have only one parameter β and are much easier to tune.

It is suggested in Sections 3.3 and 4.1 that Espresso and the von Neumann kernel with large β converge to the principal eigenvector of A, though the result does not seem to support this claim (both Simplified- and Filtered Espresso are 10 points lower than the most frequent sense baseline). The reason seems to be because Espresso and the von Neumann kernels use pointwise mutual information as a weighting factor so that the principal eigenvector of A may not always represent the most frequent sense.[7]

We also show the results of previous graph-based methods (Agirre et al., 2006), based on HyperLex (Véronis, 2004) and PageRank (Brin and Page, 1998). The experimental set-up is the same as ours in that they do not use the sense tags of training corpus to construct a co-occurrence graph, and they use the sense tags of all the S3LS training corpus for mapping senses to clusters. However, these methods have seven parameters to tune in order to achieve the best performance, and hence are difficult to optimize.

5.3 Experiment 3: Sensitivity to a Different Diffusion Factor

Figure 4 shows the performance of the von Neumann kernels with a diffusion factor β. As expected, smaller β leads to relatedness to seed instances, and larger β asymptotically converges to the HITS authority ranking (or equivalently, Simplified

[7]A similar but more extreme case is described in (Ito et al., 2005) in which the use of a normalized weight matrix M results in an unintuitive principal eigenvector.

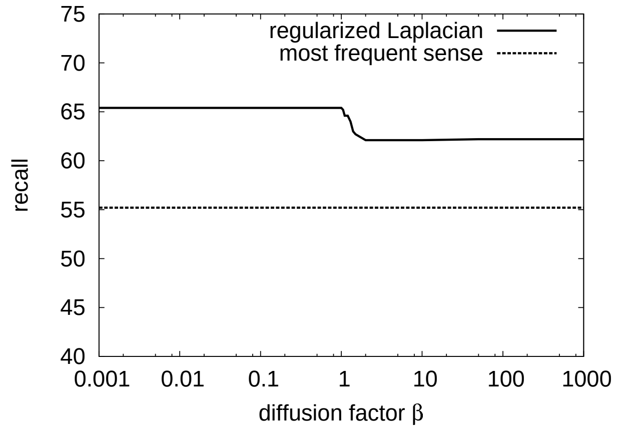


Figure 5: Recall of the regularized Laplacian with a different diffusion factor β on S3LS WSD task

Espresso).

One of the disadvantages of the von Neumann kernels over the regularized Laplacian is their sensitivity to parameter β. Figure 5 illustrates the performance of the regularized Laplacian with a diffusion factor β. The regularized Laplacian is stable for various values of β, while the von Neumann kernels change their behavior drastically depending on the value of β. However, β in the von Neumann kernels is upper-bounded by the reciprocal $1/\lambda$ of the principal eigenvalue of A, and the derivatives of kernel matrices with respect to β can be used to guide systematic calibration of β (see (Ito et al., 2005) for detail).

6 Conclusion and Future Work

This paper gives a graph-based analysis of semantic drift in Espresso-like bootstrapping algorithms. We indicate that semantic drift in bootstrapping is a parallel to topic drift in HITS. We confirm that the von Neumann kernels and the regularized Laplacian reduce semantic drift in the Senseval-3 Lexical Sample task. Our proposed methods have only one parameters and are easy to calibrate.

Beside the regularized Laplacian, many other kernels based on the eigenvalue regularization of the Laplacian matrix have been proposed in machine learning community (Kondor and Lafferty, 2002; Nadler et al., 2006; Saerens et al., 2004). One such kernel is the commute-time kernel (Saerens et al., 2004) defined as the pseudo-inverse of Laplacian. Despite having no parameters at all, it has been reported to perform well in many collaborative filtering tasks (Fouss et al., 2007). We plan to test these kernels in our task as well.

Another research topic is to investigate other semi-supervised learning techniques such as co-training (Blum and Mitchell, 1998). As we have described in this paper, self-training can be thought of a graph-based algorithm. It is also interesting to analyze how co-training is related to the proposed algorithm.

Bootstrapping algorithms have been used in many NLP applications. Two major tasks of bootstrapping are word sense disambiguation and named entity recognition. In named entity recognition task, instances are usually retained on each iteration and added to seed instance set. This seems to be because named entity recognition suffers from semantic drift more severely than word sense disambiguation. Even though this problem setting is different from ours, it needs to be verified that the graph-based approaches presented in this paper are also effective in named entity recognition.

Acknowledgements

We thank anonymous reviewers for helpful comments and for making us aware of Abney's work. The first author is partially supported by the Japan Society for Promotion of Science (JSPS), Grant-in-Aid for JSPS Fellows.

References

Steven Abney. 2004. Understanding the Yarowsky Algorithm. *Computational Linguistics*, 30(3):365–395.

Eneko Agirre, David Martínez, Oier López de Lacalle, and Aitor Soroa. 2006. Two graph-based algorithms for state-of-the-art WSD. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 585–593.

Michele Banko and Eric Brill. 2001. Scaling to Very Very Large Corpora for Natural Language Disambiguation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 26–33.

Krishna Bharat and Monika R. Henzinger. 1998. Improved algorithms for topic distillation in a hyperlinked environment. In *Proceedings of the 21st ACM SIGIR Conference*.

Avrim Blum and Tom Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-Training. In *Proceedings of the Workshop on Computational Learning Theory (COLT)*, pages 92–100. Morgan Kaufmann.

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.

Pavel Yu Chebotarev and Elena V. Shamis. 1998. On proximity measures for graph vertices. *Automation and Remote Control*, 59(10):1443–1459.

Michael Collins and Yoram Singer. 1999. Unsupervised Models for Named Entity Classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–110.

James R. Curran, Tara Murphy, and Bernhard Scholz. 2007. Minimising semantic drift with Mutual Exclusion Bootstrapping. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, pages 172–180.

François Fouss, Luh Yen, Pierr Dupont, and Marco Saerens. 2007. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):355–369.

Marti Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of the Fourteenth International Conference on Computational Linguistics*, pages 539–545.

Takahiko Ito, Masashi Shimbo, Taku Kudo, and Yuji Matsumoto. 2005. Application of Kernels to Link Analysis. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 586–592.

Rie Johnson and Tong Zhang. 2007. On the Effectiveness of Laplacian Normalization for Graph Semi-supervised Learning. *Journal of Machine Learning Research*, 8:1489–1517.

Jaz Kandola, John Shawe-Taylor, and Nello Cristianini. 2002. Learning Semantic Similarity. In *Advances in Neural Information Processing Systems 15*, pages 657–664.

Jon Kleinberg. 1999. Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM*, 46(5):604–632.

Mamoru Komachi and Hisami Suzuki. 2008. Minimally Supervised Learning of Semantic Knowledge from Query Logs. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing*, pages 358–365.

Risi Imre Kondor and John Lafferty. 2002. Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the 19th International Conference on Machine Learning (ICML-2002)*.

Hang Li and Cong Li. 2004. Word Translation Disambiguation Using Bilingual Bootstrapping. *Computational Linguistics*, 30(1):1–22.

Boaz Nadler, Stephane Lafon, Ronald Coifman, and Ioannis Kevrekidis. 2006. Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators. *Advances in Neural Information Processing Systems 18*, pages 955–962.

Vincent Ng and Claire Cardie. 2003. Weakly Supervised Natural Language Learning Without Redundant Views. In *Proceedings of the HLT-NAACL 2003*, pages 94–101.

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 113–120.

Ellen Riloff and Rosie Jones. 1999. Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intellligence (AAAI-99)*, pages 474–479.

Marco Saerens, François Fouss, Luh Yen, and Pierre Dupont. 2004. The principal component analysis of a graph, and its relationship to spectral clustering. In *Proceedings of European Conference on Machine Learning (ECML 2004)*, pages 371–383. Springer.

Heinrich Schütze. 1998. Automatic Word Sense Discrimination. *Computational Linguistics*, 24(1):97–123.

Alex J. Smola and Risi Imre Kondor. 2003. Kernels and Regularization of Graphs. In *Proceedings of the 16th Annual Conference on Learning Theory*, pages 144–158.

Jean Véronis. 2004. HyperLex: Lexical Cartography for Information Retrieval. *Computer Speech & Language*, 18(3):223–252.

David Yarowsky. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196.

The Linguistic Structure of English Web-Search Queries

Cory Barr and Rosie Jones
Yahoo! Inc.
701 First Ave
Sunnyvale, CA 94089
barrc,jonesr@yahoo-inc.com

Moira Regelson
Perfect Market, Inc.
Pasadena, CA 91103
mregelson@perfectmarket.com

Abstract

Web-search queries are known to be short, but little else is known about their structure. In this paper we investigate the applicability of part-of-speech tagging to typical English-language web search-engine queries and the potential value of these tags for improving search results. We begin by identifying a set of part-of-speech tags suitable for search queries and quantifying their occurrence. We find that proper-nouns constitute 40% of query terms, and proper nouns and nouns together constitute over 70% of query terms. We also show that the majority of queries are noun-phrases, not unstructured collections of terms. We then use a set of queries manually labeled with these tags to train a Brill tagger and evaluate its performance. In addition, we investigate classification of search queries into grammatical classes based on the syntax of part-of-speech tag sequences. We also conduct preliminary investigative experiments into the practical applicability of leveraging query-trained part-of-speech taggers for information-retrieval tasks. In particular, we show that part-of-speech information can be a significant feature in machine-learned search-result relevance. These experiments also include the potential use of the tagger in selecting words for omission or substitution in query reformulation, actions which can improve recall. We conclude that training a part-of-speech tagger on labeled corpora of queries significantly outperforms taggers based on traditional corpora, and leveraging the unique linguistic structure of web-search queries can improve search experience.

1 Introduction

Web-search queries are widely acknowledged to be short (2.8 words (Spink et al., 2002)) and to be frequently reformulated, but little else is understood about their grammatical structure. Since search queries are a fundamental part of the information retrieval task, it is essential that we interpret them correctly. However, the variable forms queries take complicate interpretation significantly. We hypothesize that elucidating the grammatical structure of search queries would be highly beneficial for the associated information retrieval task.

Previous work with queries (Allan and Raghavan, 2002) considered that short queries may be ambiguous in their part of speech and that different documents are relevant depending on how this ambiguity is resolved. For example, the word "boat" in a query may be intended as subject of a verb, object of a verb, or as a verb, with each case reflecting a distinct intent. To distinguish between the possibilities, Allan and Raghavan (Allan and Raghavan, 2002) propose eliciting feedback from the user by showing them possible contexts for the query terms.

In addition to disambiguating query terms for retrieval of suitable documents, part-of-speech tagging can help increase recall by facilitating query reformulation. Zukerman and Raskutti (Zukerman and Raskutti, 2002) part-of-speech tag well-formed questions, and use the part-of-speech tags to substitute synonyms for the content words.

Several authors have leveraged part-of-speech tagging towards improved index construction for information retrieval through part-of-speech-based weighting schemas and stopword detection (Crestani et al., 1998), (Chowdhury and McCabe, 2000), (Dincer and Karaoglan, 2004). Their experiments show degrees of success. Recently, along with term weighting, Lioma has been using part-of-speech n-grams for noise and content detection in indexes (Lioma, 2008). Our study differs from these in that linguistic and part-of-speech focus is almost exclusively placed on queries as opposed to the indexed documents, reflecting our opinion that queries exhibit their own partially predictable and unique linguistic structure different from that of the natural language of indexed documents. Similarly, (Strzalkowski et al., 1998) added a layer of natural language processing using part-of-speech tags and syntactical parsing to the common statistical information-retrieval framework, much like experiments detailed in sections 4 and 5. Our system differs in that our syntactic parsing system was applied to web-search queries and uses rules derived from the observed linguistic structure of queries as opposed to natural-language corpora. By focusing on the part-of-speech distribution and syntactic structure of queries over tagged indexed documents, with a simple bijection mapping our query tags to other tag sets, our system offers a complementary approach that can be used in tandem with the techniques referenced above.

Lima and Pederson (de Lima and Pederson, 1999) conducted related work in which part-of-speech tagging using morphological analysis was used as a preprocessing step for labeling tokens of web-search queries before being parse by a probabilistic context-free grammar tuned to query syntax. We believe this technique and others relying on part-of-speech tagging of queries could benefit from using a query-trained tagger prior to deeper linguistic analysis.

Pasca (Pasca, 2007) showed that queries can be used as a linguistic resource for discovering named entities. In this paper we show that the majority of query terms are proper nouns, and the majority of queries are noun-phrases, which may explain the success of this data source for named-entity discovery.

In this work, we use metrics that assume a unique correct part-of-speech tagging for each query, implicitly addressing the disambiguation issue through inter-annotator-agreement scores and tagger generalization error. To identify these tags, we first analyze the different general forms of queries. In Section 2 we determine a suitable set of part-of-speech labels for use with search queries. We then use manually labeled query data to train a tagger and evaluate its performance relative to one trained on the Brown corpus in Section 3. We make observations about the syntactic structure of web-search queries in Section 4, showing that the majority (70%) of queries are noun-phrases, in contrast with the commonly held belief that queries consist of unstructured collections of terms. Finally, we examine the potential use of tagging in the tasks of search relevance evaluation and query reformulation in Section 5.

2 Data

We sampled queries from the Yahoo! search engine recorded in August 2006. Queries were systematically lower-cased and white-spaced normalized. We removed any query containing a non-ASCII character. Queries were then passed through a high-precision proprietary query spelling corrector, followed by the Penn Treebank tokenizer. No other normalization was carried out. Despite Penn-tokenization, queries were typical in their average length (Jansen et al., 2000). We sampled 3,283 queries from our dataset to label, for a total of 2,508 unique queries comprised of 8,423 individual tokens.

2.1 Inter-rater Agreement

The sparse textual information in search queries presents difficulties beyond standard corpora, not only for part-of-speech tagging software but also for human labelers. To quantify the level of these difficulties we measured inter-rater agreement on a set of 100 queries labeled by each editor. Since one labeler annotated 84.4% of the queries, we used a non-standard metric to determine agreement. One hundred queries were selected at random from each of our secondary labelers. Our primary labeler then re-labeled these queries. Accuracy was then calculated as a weighted average, specifically the mean of the agreement between our primary labeler and secondary labelers, weighted by the number of queries contributed by each secondary labeler. Measuring agreement with respect to the individual part-of-speech tag for each token, our corpus has an inter-rater agreement of 79.3%. If we require agreement between all tokens in a query, agreement falls to 65.4%. Using Cohen's kappa coefficient, we have that token-level agreement is a somewhat low 0.714 and query-level agreement is an even lower 0.641.

We attempted to accurately quantify token-level ambiguity in queries by examining queries where chosen labels differ. An author-labeler examined conflicting labels and made a decision whether the difference was due to error or genuine ambiguity. Error can be a result of accidentally selecting the wrong label, linguistic misunderstanding (e.g., "chatting" labeled as a verb or gerund), or lack of consensus between editors (e.g., model numbers could be nouns, proper nouns, or even numbers). Examples of genuinely ambiguous queries include "download" and "rent," both of which could be a noun or verb. Another major source of genuine token-level ambiguity comes from strings of proper nouns. For example, some editors considered "stillwater chamber of commerce" one entity and hence four proper-noun tokens while others considered only the first token a proper noun. Of the 99 conflicting token labels in our queries used to measure inter-annotator agreement, 69 were judged due to genuine ambiguity. This left us with a metric indicating query ambiguity accounts for 69.7% of labeling error.

2.2 Tags for Part-of-Speech Tagging Queries

In preliminary labeling experiments we found many standard part-of-speech tags to be extremely rare in web-search queries. Adding them to the set of possible tags made labeling more difficult without adding any necessary resolution. In Table 1 we give the set of tags we used for labeling. In general, part-of-speech tags are defined according to the distributional behavior of the corresponding parts of speech.

Our tag set differs dramatically from the Brown or Penn tag sets. Perhaps most noticeably, the sizes of the tag sets are radically different. The Brown tag set contains roughly 90 tags. In addition, several tags can be appended with additional symbols to indicate negation, genitives, etc. Our tag set contains just 19 unique classes.

Tag	Example	Count (%)
proper-noun	texas	3384 (40.2%)
noun	pictures	2601 (30.9%)
adjective	big	599 (7.1%)
URI	ebay.com	495 (5.9%)
preposition	in	310 (3.7%)
unknown	y	208 (2.5%)
verb	get	198 (2.4%)
other	conference06-07	174 (2.1%)
comma	,	72 (0.9%)
gerund	running	69 (0.8%)
number	473	67 (0.8%)
conjunction	and	65 (0.8%)
determiner	the	56 (0.7%)
pronoun	she	53 (0.6%)
adverb	quickly	28 (0.3%)
possessive	's	19 (0.2%)
symbol	(	18 (0.2%)
sentence-ender	?	5 (0.1%)
not	n't	2 (0.0%)

Table 1: Tags used for labeling part-of-speech in web-search queries.

Our contrasting tag sets reflect an extremely different use of the English language and corresponding part-of-speech distribution. For example, the Brown tag set contains unique tags for 35 types of verbs. We use a single label to indicate all cases of verbs. However, the corpora the Brown tag set was designed for consists primarily of complete, natural-language sentences. Essentially, every sentence contains at least one verb. In contrast, a verb of any type accounts for only 2.35% of our tags. Similarly, the Brown corpus contains labels for 15 types of determiners. This class makes up just 0.66% of our data.

Our most common tag is the proper noun, which constitutes 40% of all query terms, and proper nouns and nouns together constitute 71% of query terms. In the Brown corpus, by contrast, the most common tag, noun, constitutes about 13% of terms. Thus the distribution of tag types in queries is quite different from typical edited and published texts, and in particular, proper nouns are more common than regular nouns.

2.3 Capitalization in Query Data

Although we have chosen to work with lowercase data, web search queries sometimes contain capi-

Use of Capitals	Count	%	Example
Proper-nouns capitalized	48	47%	list of Filipino riddles
Query-Initial-Caps	10	10%	Nautical map
Init-Caps + Proper-Nouns	7	7%	Condos in Yonkers
Acronym	4	4%	location by IP address
Total standard capitalization	69	67%	
All-caps	26	25%	FAX NUMBER FOR ALLEN CANNING CO
Each word capitalized	6	6%	Direct Selling
Mixed	2	2%	SONGS OF MEDONA music feature:audio
Total non-standard capitalization	34	33%	

Table 2: Ways capitalization is used in web-search queries.

talization information. Since capitalization is frequently used in other corpora to identify proper nouns, we reviewed its use in web-search queries. We found that the use of capitalization is inconsistent. On a sample of 290,122 queries from August 2006 only 16.8% contained some capitalization, with 3.9% of these all-caps. To review the use of capitalization, we hand-labeled 103 queries containing capital letters (Table 2).

Neither all-lowercase (83.2%) nor all-caps (3.9%) queries can provide us with any part-of-speech clues. But we would like to understand the use of capitalization in queries with varied case. In particular, how frequently does first-letter capitalization indicate a proper noun? We manually part-of-speech tagged 75 mixed-case queries, which contained 289 tokens, 148 of which were proper nouns. The baseline fraction of proper nouns in this sample is thus 51% (higher than the overall background of 40.2%). A total of 176 tokens were capitalized, 125 of them proper nouns. Proper nouns thus made up 73.3% of capitalized tokens, which is larger than the background occurrence of proper nouns. We can conclude from this that capitalization in a mixed-case query is a fair indicator that a word is a proper noun. However, the great majority of queries contain no informative capitalization, so the great majority of proper nouns in search queries must be uncapitalized. We cannot, therefore, rely on capitalization to identify proper nouns.

With this knowledge of the infrequent use of capital letters in search queries in mind, we will examine the effects of ignoring or using a query's capitalization for part-of-speech tagging in Section 3.4.2.

3 Tagger Accuracy on Search Queries

To investigate automation of the tagging process, we trained taggers on our manually labeled query set. We used 10-fold cross-validation, with 90% of the data used for training and the remaining data used for testing. In the sections below, we used two datasets. The first consists of 1602 manually labeled queries. For the experiments in Section 3.5 we labeled additional queries, for a total of 2503 manually labeled queries.

3.1 Part-of-Speech Tagging Software

We experimented with two freely available part-of-speech taggers: The Brill Tagger (Brill, 1995) and The Stanford Tagger (Toutanova and Manning, 2000; Toutanova et al., 2003).

The Brill tagger works in two stages. The initial tagger queries a lexicon and labels each token with its most common part-of-speech tag. If the token is not in the lexicon, it labels the token with a default tag, which was "proper noun" in our case. In the second stage, the tagger applies a set of lexical rules which examine prefixes, suffixes, and infixes. The tagger may then exchange the default tag based on lexical characteristics common to particular parts of speech. After application of lexical rules, a set of contextual rules analyze surrounding tokens and their parts of speech, altering tags accordingly.

We chose to experiment primarily with the Brill tagger because of its popularity, the human-readable rules it generates, and its easily modifiable code base. In addition, the clearly defined stages and incorporation of the lexicon provide an accessible way to supply external lexicons or entity-detection routines, which could compensate for the sparse contextual information of search queries.

We also experimented with the Stanford Log-Linear Part-of-Speech Tagger, which presently holds the best published performance in the field at 96.86% on the Penn Treebank corpus. It achieves this accuracy by expanding information sources for tagging. In particular, it provides "(i) more extensive treatment of capitalization for unknown words; (ii) features for the disambiguation of the tense forms of verbs; (iii) features for disambiguating particles from prepositions and adverbs." It uses a maximum-entropy approach to handle information diversity without assuming predictor independence (Toutanova and Manning, 2000).

3.2 Baseline: Most Common Tag

With proper nouns dominating the distribution, we first considered using the accuracy of labeling all tokens "proper noun" as a baseline. In this case, we labeled 1953 of 4759 (41.0%) tokens correctly. This is a significant improvement over the accuracy of tagging all words as "noun" on the Brown corpus (approximately 13%), reflecting the frequent occurrence of proper nouns in search queries. However, to examine the grammatical structure of search queries we must demonstrate that they are not simply collections of words. With this in mind, we chose instead to use the most common part-of-speech tag for a word as a baseline. We evaluated the baseline performance on our manually labeled dataset, with URLs removed. Each token in the set was assigned its most common part of speech, according to the Brill lexicon. In this case, 4845 of 7406 tokens were tagged correctly (65.42%).

3.3 Effect of Type of Training Data

The Brill tagger software is pre-trained on the standard Wall Street Journal corpus, so the simplest possible approach is to apply it directly to the query data set. We evaluated this "out-of-the-box" performance on our 1602 manually labeled queries, after mapping tags to our reduced tag set. (Our effective training-set size is 1440 queries, since 10% were held out to measure accuracy through cross validation.) The WSJ-trained tagger labeled 2293 of 4759 (48.2%) tags correctly, a number well below the baseline performance, demonstrating that application of the contextual rules that Brill learns from the syntax of natural-language corpora has a negative effect on accuracy in the context of queries. When we re-trained Brill's tagger on a manually labeled set of queries, we saw accuracy increase to 69.7%. The data used to train the tagger therefore has a significant effect on its accuracy (Table 3). The accuracy of the tagger trained on query data is above the baseline, indicating that search queries are somewhat more than collections of words.

3.4 Improving Tagger Accuracy

We conducted several experiments in improving tagger accuracy, summarized in Table 3 and described in detail below.

3.4.1 Adding External Lexicon

With a training-set size of 1500 queries, comprising a lexicon of roughly 4500 words, it is natural to question if expanding the lexicon by incorporating external sources boosts performance. To this end, we lower-cased the lexicon of 93,696 words provided by the Brill tagger, mapped the tags to our own tag set, and merged our lexicon from queries. This experiment resulted in an accuracy of 71.1%, a 1.4% increase.

One explanation for the limited increase is that this lexicon is derived from the Brown corpus and the Penn Treebank tagging of the Wall Street Journal. These corpora are based on works published in 1961 and 1989-1992 respectively. As shown in Table 1, proper nouns dominate the distribution of search-engine queries. Many of these queries will involve recent products, celebrities, and other time-sensitive proper nouns. We speculate that Web-based information resources could be leveraged to expand the lexicon of timely proper nouns, thereby enhancing performance.

3.4.2 Experiments with Perfect Capitalization

The overall performance of the pre-trained Brill tagger on our query set may be due to its poor performance on proper nouns, our most frequent part of speech. In the WSJ newspaper training data, proper-nouns always start with a capital letter. As discussed in Section 2.3, capitalization is rare in web-search queries. To examine the effect of the missing capitalization of proper nouns, we evaluated a pre-trained Brill tagger on our previously mentioned manually labeled corpus of 1602 queries altered such that only the proper nouns were capitalized. In this case, the tagger reached an extraordinary 89.4% accuracy (Table 3). Unfortunately, the vast majority of queries do not contain capitalization information and those that do often contain misleading information. The pre-trained tagger achieved only a 45.6% accuracy on non-lowercased queries, performing even worse than on the set with no capitalization at all.

Experiment	Accuracy
Label-all-proper-noun	41.0%
WSJ-trained	48.2%
most-freq-tag-WSJ	64.4%
re-trained	69.7%
retrained + WSJ lexicon	71.1%
user capitalization	45.6%
oracle capitalization	89.4%
automatic capitalization	70.9%

Table 3: Tagging experiments on small labeled corpus. Experiments were conducted on lower-cased queries except where specifically indicated.

3.4.3 Automatic Capitalization

We saw in Section 2.3 that web searchers rarely use capitalization. We have also seen that a pre-trained Brill tagger run on queries with perfect capitalization ("oracle" capitalization) can achieve 89.4% accuracy. We now look at how performance might be affected if we used an imperfect algorithm for capitalization.

In order to attempt to capitalize the proper nouns in queries, we used a machine-learned system which searches for the query terms and examines how often they are capitalized in the search results, weighting each capitalization occurrence by various features (Bartz et al., 2008). Though the capitalization system provides 79.3% accuracy, using this system we see an only a small increase of accuracy in part-of-speech tagging at 70.9%. This system does not improve significantly over the tagger trained on the lower-cased corpus. One explanation is that capitalization information of this type could only be obtained for 81.9% of our queries. Multiplied by accuracy, this implies that roughly 81.9% * 79.3% = 65.0% of our proper nouns are correctly cased. This suggests that any technique for proper-noun detection in search-engine queries must provide over 65.0% accuracy to see any performance increase.

Finally we looked at the capitalization as input by searchers. We trained on the oracle-capitalized corpus, and tested on raw queries without normalization. We saw an accuracy of just 45.6%. Thus using the capitalization input by web searchers is misleading and actually hurts performance.

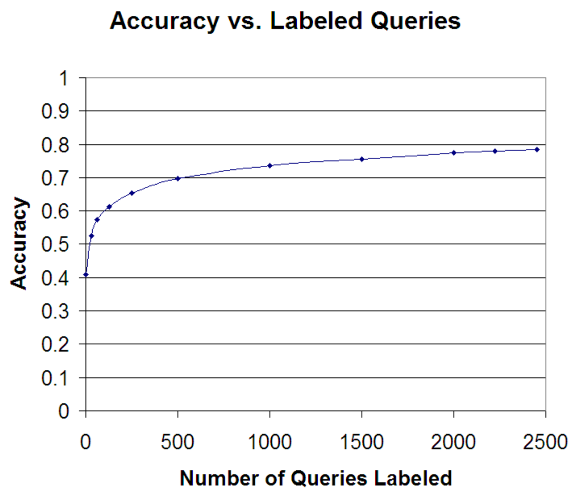


Figure 1: Brill's tagger trained on web-search queries. We see that the most significant gains in performance are with the first few hundred labeled examples, but even after 2500 examples are labeled, more labeled data continues to improve performance.

3.5 Learning Curve

It is important to understand whether tagger accuracy is limited by the small size of our manually labeled dataset. To examine the effect of dataset size, we trained Brill's tagger with increasing numbers of labeled queries and evaluated accuracy with each set size. In the interim between conducting the experiments of sections 3.1 through 3.3 and those of section 3.5, we were able to obtain 1120 new labeled queries, allowing us to extend the learning curve. With our complete corpus of 2722 labeled examples (for a cross-validated training-set size of 2450 labeled examples, URLs omitted), we see an accuracy of 78.6% on a per-token basis. We see the most significant gains in performance with the first few hundred labeled examples, but even after 2500 examples are labeled, more labeled data continues to improve performance.

3.6 Comparing Taggers to Suggest Methods for Boosting Performance

In Table 4 we see a comparison of Brill's tagger to the Stanford tagger trained on 2450 labeled queries. The 0.3% performance increase is not statistically significant. As listed in Section 3, the features the Stanford tagger adds to achieve high accuracy in traditional natural-language corpora are not in-

Tagger	Accuracy
Brill	78.6%
Stanford	78.9%

Table 4: Comparison of Brill's tagger to the Stanford tagger, on our corpus of manually annotated query logs.

formative in the domain of search-engine queries. We believe greater performance on our data will be achieved primarily through examination of common sources of inter-rater disagreement (such as consistent handling of ambiguity) and incorporation of external sources to detect proper nouns not in the lexicon.

To validate our intuition that expanding the lexicon will boost performance, we obtained a proprietary list of 7385 known trademarked terms used in the sponsored-search industry. Treating these phrases as proper nouns and adding them to the lexicon from the Wall Street Journal supplied with the Brill tagger, we see our cross validated accuracy improve to 80.2% (with a standard deviation of 1.85%), the highest score achieved in our experiments. We find it likely that incorporation of more external lexical sources will result in increased performance.

Our experiments also support our hypothesis that addressing inter-annotator agreement will boost performance. We can see this by examining the results of the experiments in section 3.3 verses section 3.5. In section 3.3, we see the accuracy on the query-trained Brill tagger is 69.7%. As mentioned, for the experiment in section 3.5, we were able to obtain 1120 new queries. Each of these newly labeled queries came from the same labeler, who believes their handling of the ambiguities inherent in search queries became more consistent over time. With the same training-set size of 1440 used in section 3.3, Figure 1 shows performance at 1440 queries is roughly 6% higher. We believe this significant improvement is a result of more consistent handling of query ambiguity obtained through labeling experience.

4 Query Grammar

The above-baseline performance of the Brill tagger trained on web-search queries suggests that web-search queries exhibit some degree of syntactical structure. With a corpus of queries labeled with part-of-speech information, we are in a position to analyze this structure and characterize the typical patterns of part-of-speech used by web searchers. To this end, we randomly sampled and manually labeled a set of 222 queries from the part-of-speech dataset used for tagger training mentioned above. Each query was labeled with a single meta-tag indicating query type. Two author-judges simultaneously labeled queries and created the set of meta-tags during much discussion, debate, and linguistic research. A list of our meta-tags and the distribution of each are provided in Table 5. We can see that queries consisting of a noun-phrase dominate the distribution of query types, in contrast with the popularly held belief that queries consist of unstructured collections of terms.

To determine how accurately a meta-tag can be determined based on part-of-speech labels, we created a grammar consisting of a set of rules to rewrite part-of-speech tags into higher-level grammatical structures. These higher-level grammatical structures are then rewritten into one of the seven classes of meta-tags seen in Table 5. Our grammar was constructed by testing the output of our rewrite rules on queries labeled with par-of-speech tags that were not part of the 222 queries sampled for meta-tag labeling. Grammar rules were revised until the failure rate on previously untested part-of-speech-labeled queries stabilized. Failure was evaluated by two means. In the first case, the grammar rules failed to parse the sequence of part-of-speech tags. In the second case, the grammar rules led to an inappropriate classification for a query type. As during the labeling phase, the two author-labelers simultaneously reached a consensus on whether a parse failed or succeeded, rendering an inter-annotator score inapplicable. The resulting grammar was then tested on the 222 queries with query-type meta-tags.

Our rules function much like production rules in context-free grammars. As an example, the two-tag sequence "determiner noun" will be rewritten as "noun phrase." This in turn could be re-written into a larger structure, which will then be rewritten into a meta-tag of query type. The primary difference between a context-free grammar or probabilistic context-free grammar (such as that employed by Lima and Pederson (de Lima and Pederson, 1999))

Query Gramm. Type	Example	Freq (%)
noun-phrase	free mp3s	155 (69.8%)
URI	http:answers.yahoo.com/	24 (10.8%)
word salad	mp3s free	19 (8.1%)
other-query	florida elementary reading conference2006-2007	15 (6.8%)
unknown	nama-nama calon praja ipdn	6 (2.7%)
verb-phrase	download free mp3s	3 (1.4%)
question	where can I download free mp3s	1 (0.45%)

Table 5: Typical grammatical forms of queries used by web searchers, with distribution based on a sample of 222 hand-labeled queries.

and our grammar is that our rules are applied iteratively as opposed to recursively. As such, our grammar yields a single parse for each input.

Some of our rules reflect the telegraphic nature of web queries. For example, it is much more common to see an abbreviated noun-phrase consisting of adjective-noun, than one consisting of determiner-adjective-noun.

Examining the Table 5, we see that just labeling a query "noun-phrase" results in an accuracy of 69.8%. Our grammar boosted this high baseline by 14% to yield an final accuracy result of 83.3% at labeling queries with their correct meta-type. These meta-types could be useful in deciding how to handle a query. Further enhancements to the grammar would likely yield a performance increase. However, we feel accuracy is currently high enough to continue with experiments towards application of leveraging grammar-deduced query types for information retrieval.

We can think of some of these meta-types as elided sentences. For example, the noun-phrase queries could be interpreted as requests of the form "how can I obtain X" or "where can I get information on X", while the verb-phrase queries are requests of the form "I would like to DO-X".

5 Applications of Part-of-Speech Tagging

Since search queries are part of an information retrieval task, we would like to demonstrate that part-of-speech tagging can assist with that task. We conducted two experiments with a large-scale machine-learned web-search ranking system. In addition, we considered the applicability of part-of-speech tags to the question of query reformulation.

5.1 Web Search Ranking

We worked with a proprietary experimental testbed in which features for predicting the relevance of a query to a document can be tested in a machine-learning framework. Features can take a wide variety of forms (boolean, real-valued, relational) and apply to a variety of scopes (the page, the query, or the combination). These features are evaluated against editorial judgements and ranked according to their significance in improving the relevance of results. We evaluated two part-of-speech tag-based features in this testbed.

The first experiment involved a simple query-level feature indicating whether the query contained a noun or a proper noun. This feature was evaluated on thousands of queries for the test. At the conclusion of the test, this feature was found to be in the top 13% of model features, ranked in order of significance. We believe this significance represents the importance of recognizing the presence of a noun in a query and, of course, matching it. Within this experimental testbed a statistically significant improvement of information-retrieval effectiveness is notoriously difficult to attain. We did not see a significant improvement in this metric. However, we feel that our feature's high ranking warrants reporting and hints at a potentially genuine boost in retrieval performance in a system less feature-rich.

The second experiment was more involved and reflected more of our intuition about the likely application of part-of-speech tagging to the improvement of search results. In this experiment, we part-of-speech tagged both queries and documents. Documents were tagged with a conventionally trained Brill tagger with the resulting Penn-style tags mapped to our tag set. Many thousands of query-document pairs were processed in this manner. The feature was based on the percent of times the part-of-speech tag of a word in the query matched the part-of-speech tag of the same word in the document. This feature was ranked in the top 12% by significance, though we again saw no statistically significant increase in overall retrieval performance.

5.2 Query Reformulation

We considered the application of part-of-speech tagging to the problem of query reformulation, in which

Part-of-speech	p(subst)	subst / seen
Number	0.49	148 / 302
Adjective	0.46	2877 / 6299
Noun	0.42	15038 / 35515
Proper noun	0.39	21478 / 55331
Gerund	0.37	112 / 300
Verb	0.31	1769 / 5718
Pronoun	0.23	300 / 1319
Conjunction	0.18	85 / 464
Adverb	0.13	105 / 790
Determiner	0.10	22 / 219
Preposition	0.08	369 / 4574
Possessive	0.08	25 / 330
Not	0.03	1 / 32
Symbol	0.02	16 / 879
Other	0.02	78 / 3294
Sentence-ender	0.01	3 / 234
Comma	0.00	4 / 991

Table 6: Probability of a word being reformulated from one query to the next, by part-of-speech tag. While proper-nouns are the most frequent tag in our corpus, adjectives are more frequently reformulated, reflecting the fact that the proper nouns carry the core meaning of the query.

a single word in the query is altered within the same user session. We used a set of automatically tagged queries to calculate change probabilities of each word by part-of-speech tag and the results are shown in Table 6.

The type of word most likely to be reformulated is "number." Examples included changing a year ("most popular baby names 2007" → "most popular baby names 2008"), while others included model, version and edition numbers ("harry potter 6" → "harry potter 7") most likely indicating that the user is looking at variants on a theme, or correcting their search need. Typically a number is a modifier of the core search meaning. The next most commonly changed type was "adjective," perhaps indicating that adjectives can be used to refine, but not fundamentally alter, the search intent. Nouns and proper nouns are the next most commonly modified types, perhaps reflecting user modification of their search need, refining the types of documents retrieved. Other parts of speech are relatively seldom modified, perhaps indicating that they are not viewed as having a large impact on the documents retrieved.

We can see from the impact of the search engine ranking features and from the table of query reformulation likelihood that making use of the grammatical structure of search queries can have an impact on result relevance. It can also assist with tasks associated with improving recall, such as query reformulation.

6 Conclusion

We have quantified, through a lexicostatistical analysis, fundamental differences between the natural language used in standard English-language corpora and English search-engine queries. These differences include reduced granularity in part-of-speech classes as well as the dominance of the noun classes in queries at the expense of classes such as verbs frequently found in traditional corpora. In addition, we have demonstrated the poor performance of taggers trained on traditional corpora when applied to search-engine queries, and how this poor performance can be overcome through query-based corpora. We have suggested that greater improvement can be achieved by proper-noun detection through incorporation of external lexicons or entity detection. Finally, in preliminary investigations into applications of our findings, we have shown that query part-of-speech tagging can be used to create significant features for improving the relevance of web search results and may assist with query reformulation. Improvements in accuracy can only increase the value of POS information for these applications. We believe that query grammar can be further exploited to increase query understanding and that this understanding can improve the overall search experience.

References

James Allan and Hema Raghavan. 2002. Using part-of-speech patterns to reduce query ambiguity. In *Proceedings of SIGIR*, pages 307–314.

Kevin Bartz, Cory Barr, and Adil Aijaz. 2008. Natural language generation in sponsored-search advertisements. In *Proceedings of the 9th ACM Conference on Electronic Commerce*, pages 1–9, Chicago, Illinois.

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

Abdur Chowdhury and M. Catherine McCabe. 2000. Improving information retrieval systems using part of speech tagging.

Fabio Crestani, Mark Sanderson, and Mounia Lalmas. 1998. Short queries, natural language and spoken document retrieval: Experiments at glasgow university. In *Proceedings of the Sixth Text Retrieval Conference (TREC-6)*, pages 667–686.

Erika F. de Lima and Jan O. Pederson. 1999. Phrase recognition and expansion for short, precision-biased queries based on a query log. In *Annual ACM Conference on Research and Development in Information Retrieval Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 145–152, Berkeley, California.

Bekir Taner Dincer and Bahar Karaoglan. 2004. The effect of part-of-speech tagging on ir performance for turkish. pages 771–778.

Bernard J. Jansen, Amanda Spink, and Tefko Saracevic. 2000. Real life, real users, and real needs: a study and analysis of user queries on the web. *Information Processing and Management*, 36(2):207–227.

Christina Amalia Lioma. 2008. *Part of speech N-grams for information retrieval*. Ph.D. thesis, University of Glasgow, Glasgow, Scotland, UK.

Marius Pasca. 2007. Weakly-supervised discovery of named entities using web search queries. In *CIKM*, pages 683–690.

Amanda Spink, B. J. Jansen, D. Wolfram, and T. Saracevic. 2002. From e-sex to e-commerce: Web search changes. *IEEE Computer*, 35(3):107–109.

Tomek Strzalkowski, Jose Perez Carballo, and Mihnea Marinescu. 1998. Natural language information retrieval: Trec-3 report. In *Proceedings of the Sixth Text Retrieval Conference (TREC-6)*, page 39.

Kristina Toutanova and Christopher D. Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL*, pages 252–259.

Ingrid Zukerman and Bhavani Raskutti. 2002. Lexical query paraphrasing for document retrieval. In *COLING*, pages 1177–1183, Taipei, Taiwan.

Mining and Modeling Relations between Formal and Informal Chinese Phrases from Web Corpora

Zhifei Li and **David Yarowsky**
Department of Computer Science and Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD 21218, USA
zhifei.work@gmail.com and yarowsky@cs.jhu.edu

Abstract

We present a novel method for discovering and modeling the relationship between informal Chinese expressions (including colloquialisms and instant-messaging slang) and their formal equivalents. Specifically, we proposed a bootstrapping procedure to identify a list of candidate informal phrases in web corpora. Given an informal phrase, we retrieve contextual instances from the web using a search engine, generate hypotheses of formal equivalents via this data, and rank the hypotheses using a conditional log-linear model. In the log-linear model, we incorporate as feature functions both rule-based intuitions and *data co-occurrence* phenomena (either as an explicit or indirect definition, or through formal/informal usages occurring in free variation in a discourse). We test our system on manually collected test examples, and find that the (*formal-informal*) relationship discovery and extraction process using our method achieves an average 1-best precision of 62%. Given the ubiquity of informal conversational style on the internet, this work has clear applications for text normalization in text-processing systems including machine translation aspiring to broad coverage.

1 Introduction

Informal text (e.g., newsgroups, online chat, blogs, etc.) is the majority of all text appearing on the Internet. Informal text tends to have very different style from formal text (e.g., newswire, magazine, etc.). In particular, they are different in vocabulary, syntactic structure, semantic interpretation, discourse structure, and so on. On the other hand, certain relations exist between the informal and formal text, and informal text often has a viable formal equivalent. Table 1 shows several naturally occurring examples of informal expressions in Chinese, and Table 2 provides a more detailed inventory and characterization of this phenomena[1]. The first example of informal phrase "88" is used very often in Chinese on-line chat when a person wants to say "bye-bye" to the other person. This can be explained as follows. In Chinese, the standard equivalent to "bye-bye" is "拜拜" whose *PinYin* is "BaiBai". Coincidentally, the *PinYin* of "88" is "BaBa". Because "BaBa" and "BaiBai" are near homophones, people often use "88" to represent "拜拜", either for input convenience or just for fun. The other relations in Table 1 are formed due to similar processes as will be described later.

Formal	**Informal**
拜拜 (BaiBai)[bye-bye]	88 (BaBa)
喜欢 (XiHuan)[like]	稀饭 (XiFan)[gruel]
哥哥 (GeGe)[elder brother]	GG
歌迷 (GeMi)[fans]	粉丝 (FenSi)[a food]

Table 1: Example Chinese *Formal-informal* Relations. The *PinYin* pronunciation is in parentheses and an optional literal gloss is in brackets.

Due to the often substantial divergence between

[1]For clarity, we represent Chinese words in the format: Chinese characters (optional *PinYin* equivalent in parentheses and optional English gloss in brackets).

informal and formal text, a text-processing system trained on formal text does not typically work well on informal genres. For example, in a machine translation system (Koehn et al., 2007), if the bilingual training data does not contain the word "稀饭" (the second example in Table 1), it leaves the word untranslated. On the other hand, if the word "稀饭" does appear in the training data but it has only a translation "gruel" as that is the meaning in the formal text, the translation system may wrongly translate "稀饭" into "gruel" for the *informal* text where the word "稀饭" is more likely to mean "like". Therefore, as a text-normalization step, it is desirable to transform the informal text into its standard formal equivalent before feeding it into a general-purpose text-processing system. Unfortunately, there are many processes for generating informal expressions in common use today. Such transformations are highly flexible/diverse, and new phrases are invented on the Internet every day due to major news events, popular movies, TV shows, radio talks, political activities, and so on. Therefore, it is of great interest to have a *data-driven* method that can *automatically* find the relations between informal and formal expressions.

In this paper, we present a novel method for discovering and modeling the relationship between informal Chinese expressions found in web corpora and their formal equivalents. Specifically, we implement a bootstrapping procedure to identify a list of candidate informal phrases. Given an individual informal phrase, we retrieve contextual instances from the web using a search engine (in this case, *www.baidu.com*), generate hypotheses of formal equivalents via this data, and rank the hypotheses using a conditional log-linear model. In the log-linear model, we incorporate as feature functions both rule-based intuitions and *data co-occurrence* phenomena (either as an explicit or indirect definition, or through formal/informal usages occurring in free variation in a discourse). We test our system on manually collected test examples[2], and find that the (*formal-informal*) relationship discovery and extraction process using our method achieves an average precision of more than 60%. This work has applications for text normalization in many general-purpose text-processing tasks, e.g., machine translation.

To the best of our knowledge, our work is the first published machine-learning approach to productively model the broad types of relationships between informal and formal expressions in Chinese using web corpora.

2 Formal to Informal: Phenomena and Examples

In this section, we describe the phenomena and provide examples of the relations between formal and informal expressions in Chinese (we refer to the relation as *formal-informal* phrases hereafter, even in the case of single-word expressions). We manually collected 908 *formal-informal* relations, and classified these relations into four categories. We collected these pairs by investigating multiple webpages where the *formal-informal* relations are manually compiled, and then merged these seed relations and removed duplicates. In this way, the 908 examples should give good coverage on the typical categories in the *formal-informal* relations. Also, the distribution of the categories found in the 908 examples should be representative of the actual distribution of the *formal-informal* relations occurring in the real text. Table 2 presents these categories and examples in each category. In the last column, the table also shows the relative frequency of each category, computed based on the 908 examples. Recall that we represent Chinese words in the format: Chinese characters (optional *PinYin* equivalent in parentheses and optional English gloss in brackets).

2.1 Homophone

In general, a homophone is a word that is pronounced the same as another word but differs in meaning and/or written-form. Here, we use the word "homophone" in a loose way. In particular, we refer an informal phrase as a homophone of a formal phrase if its pronunciation is the same or *similar* to the formal phrase. In the three examples belonging to the homophone category in Table 2, the first example is a true homophone, while the other two are loose homophones. The third example represents a major sub-class where the informal phrase is a *number* (e.g., 88).

[2]The training and test examples are freely available at http://www.cs.jhu.edu/∼zfli.

Category	**Formal**	**Informal**	**%**
Homophone	版主 (BanZhu) [system administrator]	斑竹 (BanZhu) [bamboo]	4.2
	喜欢 (XiHuan)[like]	稀饭 (XiFan)[gruel]	4.4
	拜拜 (BaiBai)[bye-bye]	88 (BaBa)	21
Abbreviation	美国军队 (MeiGuoJunDui)[american army]	美军 (MeiJun)[american army]	3.8
Acronym	哥哥 (GeGe)[elder brother]	GG	12.3
	女朋友 (N*ü*PengYou)[girl friend]	GF	7.2
Transliteration	歌迷 (GeMi)[fans]	粉丝 (FenSi)[a Chinese food]	2.3
	谢谢 (XieXie)[thank you]	3Q (SanQiu)	
Others	希拉里粉丝 (XiLaLiFenSi)[fans of Hilary]	稀饭 (XiFan)[gruel]	44.8
	奥巴马粉丝 (AoBaMaFenSi)[fans of Obama]	藕粉 (OuFen)[a food]	
	超强 (ChaoQiang)[super strong]	走召弓虽 (ZouZhaoGongXu)	

Table 2: Chinese *Formal-informal* Relations: Categories and Examples. Literal glosses in brackets.

For illustrative purposes, we can present the *transformation path* showing how the informal phrase is obtained from the formal phrase. In particular, the *transformation path* for this category is “Formal → PinYin → Informal (similar or same PinYin as the formal phrase)”.

2.2 Abbreviation and Acronym

A Chinese *abbreviation* of a formal phrase is obtained by selecting *one or more* characters from this formal phrase, and the selected characters can be at *any* position in the formal phrase (Li and Yarowsky, 2008; Lee, 2005; Yin, 1999). In comparison, an *acronym* is a special form of abbreviation, where only the first character of each word in the formal phrase is selected to form the informal phrase. Table 2 presents three examples belonging to this category. While the first example is an abbreviation, and the other two examples are acronyms.

The *transformation path* for the second example is “Formal → PinYin → Acronym”, and the *transformation path* for the third example is “Formal → English → Acronym”. Clearly, they differ in whether *PinYin* or English is used as a bridge.

2.3 Transliteration

A transliteration is transcribing a word or text written in one writing system into another writing system. Table 2 presents examples belonging to this category. In the first example, the Chinese informal phrase “粉丝 (FenSi)[a Chinese food]” can be thought as a transliteration of the English phase “fans” as the pronunciation of “fans” is quite similar to the *PinYin* “FenSi”.

The *transformation path* for this category is “Formal → English → Chinese Transliteration”.

2.4 Others

Due to the inherently informal and flexible nature of expressions in informal genre, the formation of an informal phrase can be very complex or ad-hoc. For example, an informal phrase can be generated by applying the above transformation rules *jointly*. More importantly, many relations cannot be described using a simple set of rules. Table 2 presents three such examples, where the first two examples are generated by applying rules *jointly* and the third example is created by decomposing the Chinese characters in the formal form. The statistics collected from the 904 examples tells us that about 45% of the relations belonging to this category. This motivates us to use a *data-driven* method to automatically discover the relations between informal and formal phrases.

3 Data Co-occurrence

In natural language, related words tend to appear together (i.e., co-occurrence). For example, *Bill Gates* tends to appear together with *Microsoft* more often than expected by chance. Such co-occurrence may imply the existence of a relationship, and is exploited in *formal-informal* relation discovery under different conditions.

3.1 Data Co-occurrence in Definitions

In general, for many informal phrases in popular use, there is likely to be an explicit definition somewhere that provides or paraphrases its meaning for an unfamiliar audience. People have created dedicated definition web-pages to explain the relations between formal and informal phrases. For example, the first example in Table 3 is commonly explained in many dedicated definition web-pages on the Internet. On the other hand, in some formal text (e.g., research papers), people tend to define the informal phrase before it is used frequently in the later part of the text. The second example of Table 3 illustrates this phenomena. Clearly, the definition text normally contains salient patterns. For example, the first example follows the "*informal*是*formal*的意思" definition pattern, while the second example follows the pattern "*formal* (*informal*)". This gives us a reliable way to seed and bootstrap a list of informal phrases as will be discussed in Section 4.1.

Relation	**Definition Text**
(女朋友, GF)	GF是女朋友的意思。
(世界卫生组织, 世卫)	香港的食水采用世界卫生组织 (世卫)饮用水水质指引……

Table 3: Data Co-occurrence in Definitions

3.2 Data Co-occurrence in Online Chat

Informal phrases appear in online chat very often for input convenience or just for fun. Since different people may have different ways or traditions to express semantically-equivalent phrases, one may find many nearby *data co-occurrence* examples in chat text. For example, in Table 4, after a series of message exchanges, person A wants to end the conversation and types "拜拜" (meaning "bye-bye"), person B later includes the same semantic content, but in a different (more or less formal) expression (e.g. "88").

	⋮
Person A:	对不起， 我要先下线了
Person A:	拜拜
Person B:	88

Table 4: Data Co-occurrence in Online Chat for Relation (拜拜, 88) meaning "bye-bye"

3.3 Data Co-occurrence in News Articles

For some *formal-informal* relations, since both of the informal and formal phrases have been used in public very often and people are normally aware of these relations, an author may use the informal and formal phrases interchangeably without bothering to explain the relations. This is particularly true in news articles for some well-known relations. Table 5 shows an example, where the abbreviation "冬奥会" (meaning "winter olympics") appears in the title and its full-form "冬季奥运会" appears in the text of the same document. In general, the relative distance between an informal phrase and its formal phrase varies. For example, they may appear in the same sentence, or in neighboring sentences.

Title	都灵**冬奥会**开幕式将激情上演
Text	新华社都灵2月9日电(记者丁莹阎涛)第20届**冬季奥运会**的开幕式将于当地时间10日晚8点在都灵奥林匹克体育场正式揭开神秘的面纱。

Table 5: Data Co-occurrence in News Article for Relation (冬季奥运会, 冬奥会) meaning "winter olympics"

4 Mining Relations between Informal and Formal Phrases from Web

In this section, we describe an approach that automatically discovers the relation between a formal phrase and an informal phrase from web corpora. Specifically, we propose a bootstrapping procedure to identify a list of candidate informal phrases. Given a target informal phrase, we retrieve a large set of of instances in context from the Web, generate candidate hypotheses (i.e, candidate formal phrases) from the data, and rank the hypotheses by using a conditional log-linear model. The log-linear model is very flexible to incorporate both the rule- and data-driven intuitions (described in Sections 2 and 3, respectively) into the model as feature functions.

4.1 Identifying Informal Phrases

Before finding the formal phrase corresponding to an informal phrase, we first need to identify informal phrases of interest. For example, one can collect informal phrases manually. However, this is too expensive as new relations between informal and formal phrases emerge every day on the Internet. Alternatively, one can employ a large amount of formal text (e.g., newswire) and informal text (e.g., Internet blogs) to derive such a list as follows. Specifically, from the informal corpus we can extract those phrases whose frequency in the informal corpus is significantly different from that in the formal corpus. However, such a list may be quite noisy, i.e., many of them are not informal phrases at all.

An alternative approach to extracting the informal phrases is to use a bootstrapping algorithm (e.g., Yarowsky (1995)). Specifically, we first manually collect a small set of example relations. Then, using these relations as a *seed set*, we extract the text *patterns* (e.g., the definition pattern showing how the informal and formal phrases co-occur in the data as discussed in Section 3.1). With these patterns, we identify many more new relations from the data and augment them into the seed set. The procedure iterates. Using such an approach, we should be able to extract a large list of *formal-informal* relations. Clearly, the list extracted in this way may be quite noisy, and thus it is important to exploit both the data- and rule-driven intuitions to rank these relations properly.

4.2 Retrieving Data from Web

Given an informal phrase, we retrieve training data from the web on the fly. Specifically, we first use a search engine to identify a set of hyper-links that point to web pages containing contexts relevant to the informal phrase, and then follow the hyper-links to download the web pages. The input to the search engine is a text query. One can simply use the informal phrase as a query. However, this may lead to a set of pages that have nothing to do with the informal phrase. For example, if we search the informal phrase “88” (the third example in Table 2) using the well-known Chinese search engine *www.baidu.com*, none of the top-10 pages are related to the informal phrase “88”. To avoid this situation, one can use a search engine that is dedicated to informal text search (e.g., *blogsearch.baidu.com*). Alternatively, one can use the general-purpose search engine but expanding the query with domain information. For example, for the informal phrase “88”, we can use a query “88 网络语言”, where “网络语言” means *internet language*.

4.3 Generating Candidate Hypotheses

Given an informal phrase, we generate a set of hypotheses which are candidate formal phrases corresponding to the informal phrase. We considered two general approaches to the generation of hypotheses.

Rule-driven Hypothesis Generation: One can use the rules described in Section 2 to generate a set of hypotheses. However, with this approach, one may generate an exponential number of hypotheses. For example, assuming the number of English words starting with a given letter is O($|V|$), we can generate O($|V|^n$) hypotheses given an *acronym* containing n letters. Another problem with this approach is that a relation between an informal phrase and a formal phrase may not be explained by a specific rule. In fact, as shown in the last row of Table 2, such relations consist of 44.8% of all corpus instances.

Data-driven Hypothesis Generation: With data retrieved from the Web, we can generate hypotheses by enumerating the frequent n-grams co-occurring with the informal phrase within certain distance. This exploits the *data co-occurrence* phenomena described in Section 3, that is, the formal phrase tends to co-occur with the informal phrase nearby in the data, for the multiple reasons described above. This can deal with the cases where the relation between an informal phrase and a formal phrase cannot be explained by a rule. However, it also suffers from the over-generation problem as in the rule-driven approach.

In this paper, we use the data-driven method to generate hypotheses, and rank the hypotheses using a conditional log-linear model that incorporates both the rule and data intuitions as feature functions.

4.4 Ranking Hypotheses: Conditional Log-linear Model

Log-linear models are known for flexible incorporation of features into the model. Each feature function reflects a hint/intuition that can be used to rank the hypotheses. In this subsection, we develop a conditional log-linear model that incorporates both the rule and data intuitions as feature functions.

4.4.1 Conditional Log-linear Model

Given an informal phrase (say x) and a candidate formal phrase (say y), the model assigns the pair a score (say $s(x, y)$), which will be used to rank the hypothesis y. The score $s(x, y)$ is a linear combination of the feature scores (say $\Phi_i(x, y)$) over a set of feature functions indexed by i. Formally,

$$s(x, y) = \sum_{i=1}^{K} \Phi_i(x, y) \times \alpha_i \tag{1}$$

where K is the number of feature functions defined and α_i is the weight assigned to the i-th feature function (i.e., Φ_i). To learn the weight vector $\vec{\alpha}$, we first define a probability measure,

$$P_{\vec{\alpha}}(y|x) = \frac{1}{Z(x, \vec{\alpha})} e^{s(x,y)} \tag{2}$$

where $Z(x, \vec{\alpha})$ is a normalization constant. Now, we define the regularized log-likelihood (LL_R) of the training data (i.e, a set of pairs of (x, y)), as follows,

$$LL_R(\vec{\alpha}) = \sum_{j=1}^{N} \log P_{\vec{\alpha}}(y_j|x_j) - \frac{||\vec{\alpha}||^2}{2\sigma^2} \tag{3}$$

where N is the number of training examples, and the regularization term $\frac{||\vec{\alpha}||^2}{2\sigma^2}$ is a Gaussian prior with a variance σ^2 (Roark et al., 2007). The optimal weight vector $\vec{\alpha}^*$ is obtained by maximizing the regularized log-likelihood (LL_R), that is,

$$\vec{\alpha}^* = \arg\max_{\vec{\alpha}} LL_R(\vec{\alpha}) \tag{4}$$

To maximize the above function, we use a limited-memory variable method (Benson and More, 2002) that is implemented in the TAO package (Benson et al., 2002) and has been shown to be very effective in various natural language processing tasks (Malouf, 2002).

During test time, the following **decision rule** is normally used to predict the optimal formal phrase y^* for a given informal phrase x,

$$y^* = \arg\max_{y} s(x, y). \tag{5}$$

4.4.2 Feature Functions

As mentioned before, we incorporate both the rule- and data-driven intuitions as feature functions in the log-linear model.

Rule-driven feature functions: Clearly, if a pair (x, y) matches the rule patterns described in Table 2, the pair has a high possibility to be a true *formal-informal* relation. To reflect this intuition, we develop several feature functions as follows.

- LD-PinYin(x, y): the *Levenshtein distance* on PinYin of x and y. The distance between two PinYin characters is weighted based on the similarity of pronunciation, for example, the weight $w(l, n)$ is smaller than the weight $w(a, z)$.
- LEN-PinYin(x, y): the difference in the number of PinYin characters between x and y.
- Is-PinYin-Acronym(x, y): is x a PinYin acronym of y? For example,
  Is-PinYin-Acronym(GG, 哥哥)=1,
  Is-PinYin-Acronym(GG, 兄弟)=0.
- Is-CN-Abbreviation(x, y): is x a Chinese abbreviation of y? For example,
  Is-CN-Abbreviation(美军,美国军队)=1,
  Is-CN-Abbreviation(美军,中国军队)=0.

Data-driven feature functions: As described in Section 3, the informal and formal phrases tends to co-occur in the data. Here, we develop several feature functions to reflect this intuition.

- n-gram co-occurrence relative frequency: we collect the n-grams that occur in the data within a window of the occurrence of the informal phrase, and compute their relative frequency as feature values. Since different orders of grams will have quite different statistics, we define 7 features in this category: 1-gram, 2-gram, 3-gram, 4-gram, 5-gram, 6to10-gram, and 11to15-gram. Note that the *order* n of a n-gram is in terms of number of Chinese characters instead of words.

- Features on a definition pattern: we have discussed definition patterns in Section 3.1. For each definition pattern, we can define a feature function saying that if the co-occurrence of x and y satisfies the definition pattern, the feature value is one, otherwise is zero.
- Features on the number of relevant web-pages: another interesting feature function can be defined as follows. For each candidate relation (x, y), we use the pair as a query to search the web, and treat the *number* of pages returned by the search engine as a feature value.[3] However, these features are quite expensive as millions of queries may need to be served.

Category	**Feature**	**Weight**
Rule-driven	LD-PinYin	0.800
	Len-PinYin	0.781
	Is-PinYin-Acronym	7.594
	Is-CN-Abbreviation	7.464
Data-driven	1-gram	14.506
	2-gram	108.193
	3-gram	82.975
	4-gram	66.872
	5-gram	42.258
	6to10-gram	21.229
	11to15-gram	0.985

Table 6: Optimal Weights in the Log-linear Model

5 Experimental Results

Recall that in Section 2 we categorize the *formal-informal* relations based on the manually collected relations. In this section, we use a subset of them for training and testing. In particular, we use 252 examples to train the log-linear model that is described in Section 4, and use 249 examples as test data to compute the precision.[4]

Table 6 shows the weights[5] learned for the various feature functions described in Section 4.4. Clearly, different feature functions get quite different weights. This is intuitive as the feature functions may differ in the scale of the feature values or in their importance in ranking the hypotheses. In fact, this shows the importance of using the log-linear model to learn the optimal weights in a principled and automatic manner, instead of manually tuning the weights in an ad-hoc way.

Tables 7-9 show the precision results for different categories as described in Section 2, using the rule-driven, data-driven, or both rule and data-driven features, respectively. In the tables, the precision corresponding to the "top-N" is computed in the following way: if the true hypothesis is among the top-N hypotheses ranked by the model, we tag the classification as correct, otherwise as wrong. Clearly, the larger the N is, the higher the precision is. Computing the top-N precision (instead of just computing the usual top-1 precision) is meaningful especially when we consider our relation extractor as an intermediate step in an end-to-end text-processing system (e.g., machine translation) since the final decision can be delayed to later stage based on more evidence. In general, our model gets quite respectably high precision for such a task (e.g., more than 60% for top-1 and more than 85% for top-100) when using both data and rule-driven features, as shown in Table 9. Moreover, the data-driven features are more helpful than the rule-driven features (e.g, 25.3% absolute improvement in 1-best precision), while the combination of these features does boost the performance of any individual feature set (e.g., 10.4% absolute improvement in 1-best precision over the case using data-driven features only).

We also carried out experiments (see Table 10) in the bootstrapping procedure described in Section 4.1. In particular, we start from a seed set having 130 relations. We identify the frequent patterns from the data retrieved from the web for these seed examples. Then, we use these patterns to identify many more new possible *formal-informal* relations. After the first iteration, we select the top 3000 pairs of relations matched by the patterns. The recall of a manually collected test set (having 750 pairs) on these 3000 pairs is around 30%, which is quite promising given the highly noisy data.

[3]Note that the number of pages relevant to a query can be easily obtained as most search engines return this number.

[4]Again, the training and test examples are freely available at http://www.cs.jhu.edu/∼zfli.

[5]Note that we do not use the features on definition patterns and on the number of relevant web pages, for efficiency.

Category		Precision (%)			
		Top-1	Top-10	Top-50	Top-100
Homophone	Same PinYin	31.6	47.4	68.4	73.7
	Similar PinYin	15.0	35.0	45.0	50.0
	Number	31.6	64.2	84.2	90.5
Abbreviation	Chinese abbreviation	11.8	35.3	41.2	41.2
Acronym	PinYin Acronym	39.3	82.1	91.1	92.9
	English Acronym	3.1	6.3	9.4	28.1
Transliteration		10.0	20.0	20.0	20.0
Average		*26.1*	*53.4*	*66.3*	*72.3*

Table 7: **Rule-driven Features only**: Precision on Chinese *Formal-informal* Relation Extraction

Category		Precision (%)			
		Top-1	Top-10	Top-50	Top-100
Homophone	Same PinYin	52.6	73.7	73.7	78.9
	Similar PinYin	45.0	65.0	75.0	75.0
	Number	66.3	86.3	94.7	96.8
Abbreviation	Chinese abbreviation	0.0	23.5	47.1	47.1
Acronym	PinYin Acronym	58.9	78.6	85.7	87.5
	English Acronym	25.0	46.9	68.6	68.8
Transliteration		50.0	50.0	50.0	50.0
Average		*51.4*	*71.1*	*81.1*	*82.7*

Table 8: **Data-driven Features only**: Precision on Chinese *Formal-informal* Relation Extraction

Category		Precision (%)			
		Top-1	Top-10	Top-50	Top-100
Homophone	Same PinYin	63.2	73.7	84.2	84.2
	Similar PinYin	40.0	60.0	70.0	80.0
	Number	81.1	91.6	95.8	96.8
Abbreviation	Chinese abbreviation	11.8	41.2	52.9	52.9
Acronym	PinYin Acronym	82.1	94.6	96.4	96.4
	English Acronym	21.9	46.9	56.3	59.4
Transliteration		20.0	40.0	50.0	50.0
Average		**61.8**	**77.1**	**83.1**	**84.7**

Table 9: **Both Data and Rule-drive Features**: Precision on Chinese *Formal-informal* Relation Extraction

Size of seed set	130
Size of candidate set	3000
Size of test set	750
Recall	**30%**

Table 10: Recall of Test Set on a Candidate Set Extracted by a *Bootstrapping* Procedure

6 Related Work

Automatically extracting the relations between full-form Chinese phrases and their abbreviations is an interesting and important task for many NLP applications (e.g., machine translation, information retrieval, etc.). Recently, Chang and Lai (2004), Lee (2005), Chang and Teng (2006), Li and Yarowsky (2008) have investigated this task. Specifically, Chang and Lai (2004) describes a hidden markov model (HMM) to model the relationship between a full-form phrase and its abbreviation, by treating the abbreviation as the *observation* and the full-form words as *states* in the model. Using a set of manually-created *full-abbreviation* relations as training data, they report experimental results on a *recognition* task (i.e., given an abbreviation, the task is to obtain its full-form, or the vice versa). Chang and Teng (2006) extends the work in Chang and Lai (2004) to automatically extract the relations between full-form phrases and their abbreviations, where both the full-form phrase and its abbreviation are not given. Clearly, the method in (Chang and Lai, 2004; Chang and Teng, 2006) is *supervised* because it requires the *full-abbreviation* relations as training data. Li and Yarowsky (2008) propose an *unsupervised* method to extract the relations between full-form phrases and their abbreviations. They exploit the *data co-occurrence* phenomena in the newswire text, as we have done in this paper. Moreover, they augment and improve a statistical machine translation by incorporating the extracted relations into the baseline translation system.

Other interesting work that addresses a similar task as ours includes the work on homophones (e.g., Lee and Chen (1997)), abbreviations with their definitions (e.g., Park and Byrd (2001)), abbreviations and acronyms in the medical domain (Pakhomov, 2002), and transliteration (e.g., (Knight and Graehl, 1998; Virga and Khudanpur, 2003; Li et al., 2004; Wu and Chang, 2007)).

While all the above work deals with the relations occurring within the *formal* text, we consider the *formal-informal* relations that occur across both formal and informal text, and we extract the relations from the web corpora, instead from just formal text. Moreover, our method is *semi-supervised* in the sense that the weights of the feature functions are tuned in a *supervised* log-linear model using a small number of seed relations while the generation and ranking of the hypotheses are *unsupervised* by exploiting the *data co-occurrence* phenomena.

7 Conclusions

In this paper, we have first presented a taxonomy of the *formal-informal* relations occurring in Chinese text. We have then proposed a novel method for discovering and modeling the relationship between informal Chinese expressions (including colloquialisms and instant-messaging slang) and their formal equivalents. Specifically, we have proposed a bootstrapping procedure to identify a list of candidate informal phrases in web corpora. Given an informal phrase, we retrieved contextual instances from the web using a search engine, generated hypotheses of formal equivalents via this data, and ranked the hypotheses using a conditional log-linear model. In the log-linear model, we incorporated as feature functions both rule-based intuitions and *data co-occurrence* phenomena (either as an explicit or indirect definition, or through formal/informal usages occurring in free variation in a discourse). We tested our system on manually collected test examples, and found that the (*formal-informal*) relationship discovery and extraction process using our method achieves an average 1-best precision of 62%. Given the ubiquity of informal conversational style on the internet, this work has clear applications for text normalization in text-processing systems including machine translation aspiring to broad coverage.

Acknowledgments

We would like to thank Yi Su, Sanjeev Khudanpur, and the anonymous reviewers for their helpful comments. This work was partially supported by the Defense Advanced Research Projects Agency's GALE program via Contract № HR0011-06-2-0001.

References

S. J. Benson, L. C. McInnes, J. J. More, and J. Sarich. 2002. Tao users manual, Technical Report ANL/MCS-TM-242-Revision 1.4, Argonne National Laboratory.

S. J. Benson and J. J. More. 2002. A limited memory variable metric method for bound constrained minimization. preprint ANL/ACSP909-0901, Argonne National Laboratory.

Jing-Shin Chang and Yu-Tso Lai. 2004. A preliminary study on probabilistic models for Chinese abbreviations. *In Proceedings of the 3rd SIGHAN Workshop on Chinese Language Processing*, Barcelona, Spain (2004),pages 9-16.

Jing-Shin Chang and Wei-Lun Teng. 2006. Mining Atomic Chinese Abbreviation Pairs: A Probabilistic Model for Single Character Word Recovery. *In Proceedings of the 5rd SIGHAN Workshop on Chinese Language Processing*, Sydney, Australia (2006), pages 17-24.

Kevin Knight and Jonathan Graehl. 1998. Machine Transliteration. *Computational Linguistics,* 24(4):599-612.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan,Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constrantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. *In Proceedings of ACL*, Demonstration Session, pages 177-180.

H.W.D Lee. 2005. A study of automatic expansion of Chinese abbreviations. MA Thesis, The University of Hong Kong.

Yue-Shi Lee and Hsin-Hsi Chen. 1997. Applying Repair Processing in Chinese Homophone Disambiguation. *In Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 57-63.

Haizhou Li, Min Zhang, and Jian Su. 2004. A joint source channel model for machine transliteration. *In Proceedings of ACL 2004*, pages 159-166.

Zhifei Li and David Yarowsky. 2008. Unsupervised Translation Induction for Chinese Abbreviations using Monolingual Corpora. *In Proceedings of ACL 2008*, pages 425-433.

R. Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. *In Proceedings of CoNLL 2002*, pages 49-55.

Serguei Pakhomov. 2002. Semi-Supervised Maximum Entropy Based Approach to Acronym and Abbreviation Normalization in Medical Texts. *In Proceedings of ACL 2002*, pages 160-167.

Youngja Park and Roy J. Byrd. 2001. Hybrid text mining for finding abbreviations and their definitions. *In Proceedings of EMNLP 2001*, pages 126-133.

Brian Roark, Murat Saraclar, and Michael Collins. 2007. Discriminative n-gram language modeling. *Computer Speech and Language*, 21(2):373-392.

Paola Virga and Sanjeev Khudanpur. 2003. Transliteration of Proper Names in Cross lingual Information Retrieval. *In Proceedings of the ACL 2003 Workshop on Multilingual and Mixed-language Named Entity Recognition.*

Jian-Cheng Wu and Jason S. Chang. 2007. Learning to Find English to Chinese Transliterations on the Web. *In Proceedings of EMNLP-CoNLL 2007*, pages 996-1004.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. *In Proceedings of ACL 1995*, pages 189-196.

Z.P. Yin. 1999. Methodologies and principles of Chinese abbreviation formation. In Language Teaching and Study, No.2 (1999) 73-82.

Unsupervised Multilingual Learning for POS Tagging

Benjamin Snyder and **Tahira Naseem** and **Jacob Eisenstein** and **Regina Barzilay**
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
77 Massachusetts Ave., Cambridge MA 02139
{bsnyder, tahira, jacobe, regina}@csail.mit.edu

Abstract

We demonstrate the effectiveness of multilingual learning for unsupervised part-of-speech tagging. The key hypothesis of multilingual learning is that by combining cues from multiple languages, the structure of each becomes more apparent. We formulate a hierarchical Bayesian model for jointly predicting bilingual streams of part-of-speech tags. The model learns language-specific features while capturing cross-lingual patterns in tag distribution for aligned words. Once the parameters of our model have been learned on bilingual parallel data, we evaluate its performance on a held-out monolingual test set. Our evaluation on six pairs of languages shows consistent and significant performance gains over a state-of-the-art monolingual baseline. For one language pair, we observe a relative reduction in error of 53%.

1 Introduction

In this paper, we explore the application of multilingual learning to part-of-speech tagging when no annotation is available. This core task has been studied in an unsupervised monolingual framework for over a decade and is still an active area of research. In this paper, we demonstrate the effectiveness of multilingual learning when applied to both closely related and distantly related language pairs. We further analyze the language features which lead to robust bilingual performance.

The fundamental idea upon which our work is based is that the patterns of ambiguity inherent in part-of-speech tag assignments differ across languages. At the lexical level, a word with part-of-speech tag ambiguity in one language may correspond to an unambiguous word in the other language. For example, the word "can" in English may function as an auxiliary verb, a noun, or a regular verb. However, each of the corresponding functions in Serbian is expressed with a distinct lexical item. Languages also differ in their patterns of structural ambiguity. For example, the presence of an article in English greatly reduces the ambiguity of the succeeding tag. In Serbian, a language without articles, this constraint is obviously absent. The key idea of multilingual learning is that by combining cues from multiple languages, the structure of each becomes more apparent.

While multilingual learning can address ambiguities in each language, it must be flexible enough to accommodate cross-lingual variations such as tag inventory and syntactic structure. As a result of such variations, two languages often select and order their tags differently even when expressing the same meaning. A key challenge of multilingual learning is to model language-specific structure while allowing information to flow between languages.

We jointly model bilingual part-of-speech tag sequences in a hierarchical Bayesian framework. For each word, we posit a hidden tag state which generates the word as well as the succeeding tag. In addition, the tags of words with common semantic or syntactic function in parallel sentences are combined into bilingual nodes representing the tag pair. These joined nodes serve as anchors that create probabilistic dependencies between the tag se-

Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, pages 1041–1050,
Honolulu, October 2008. ©2008 Association for Computational Linguistics

quences in each language. We use standard tools from machine translation to discover aligned word-pairs, and thereafter our model treats the alignments as observed data.

Our model structure allows language-specific tag inventories. Additionally, it assumes only that the tags at joined nodes are *correlated*; they need not be identical. We factor the conditional probabilities of joined nodes into two individual transition probabilities as well as a coupling probability. We define priors over the transition, emission, and coupling parameters and perform Bayesian inference using Gibbs sampling and the Metropolis-Hastings algorithm.

We evaluate our model on a parallel corpus of four languages: English, Bulgarian, Serbian, and Slovene. For each of the six language pairs, we train a bilingual model on this corpus, and evaluate it on held-out monolingual test sets. Our results show consistent improvement over a monolingual baseline for all languages and all pairings. In fact, for one language pair – Serbian and Slovene – the error is reduced by over 53%. Moreover, the multilingual model significantly reduces the gap between unsupervised and supervised performance. For instance, in the case of Slovene this gap is reduced by 71%. We also observe significant variation in the level of improvement across language pairs. We show that a cross-lingual entropy measure corresponds with the observed differentials in performance.

2 Related Work

Multilingual Learning A number of approaches for multilingual learning have focused on inducing cross-lingual structures, with applications to machine translation. Examples of such efforts include work on the induction of synchronous grammars (Wu and Wong, 1998; Chiang, 2005) and learning multilingual lexical resources (Genzel, 2005).

Another thread of work using cross-lingual links has been in word-sense disambiguation, where senses of words can be *defined* based on their translations (Brown et al., 1991; Dagan et al., 1991; Resnik and Yarowsky, 1997; Ng et al., 2003).

When annotations for a task of interest are available in a source language but are missing in the target language, the annotations can be projected across a parallel corpus (Yarowsky et al., 2000; Diab and Resnik, 2002; Padó and Lapata, 2006; Xi and Hwa, 2005). In fact, projection methods have been used to train highly accurate part-of-speech taggers (Yarowsky and Ngai, 2001; Feldman et al., 2006). In contrast, our own work assumes that annotations exist for neither language.

Finally, there has been recent work on applying unsupervised multilingual learning to morphological segmentation (Snyder and Barzilay, 2008). In this paper, we demonstrate that unsupervised multilingual learning can be successfully applied to the sentence-level task of part-of-speech tagging.

Unsupervised Part-of-Speech Tagging Since the work of Merialdo (1994), the HMM has been the model of choice for unsupervised tagging (Banko and Moore, 2004). Recent advances in these approaches include the use of a fully Bayesian HMM (Johnson, 2007; Goldwater and Griffiths, 2007). In very recent work, Toutanova and Johnson (2008) depart from this framework and propose an LDA-based generative model that groups words through a latent layer of ambiguity classes thereby leveraging morphological features. In addition, a number of approaches have focused on developing discriminative approaches for unsupervised and semi-supervised tagging (Smith and Eisner, 2005; Haghighi and Klein, 2006).

Our focus is on developing a simple model that effectively incorporates multilingual evidence. We view this direction as orthogonal to refining monolingual tagging models for any particular language.

3 Model

We propose a bilingual model for unsupervised part-of-speech tagging that jointly tags parallel streams of text in two languages. Once the parameters have been learned using an untagged bilingual parallel text, the model is applied to a held-out monolingual test set.

Our key hypothesis is that the patterns of ambiguity found in each language at the part-of-speech level will differ in systematic ways; by considering multiple language simultaneously, the total inherent ambiguity can be reduced in each language. The model is designed to permit information to flow across the

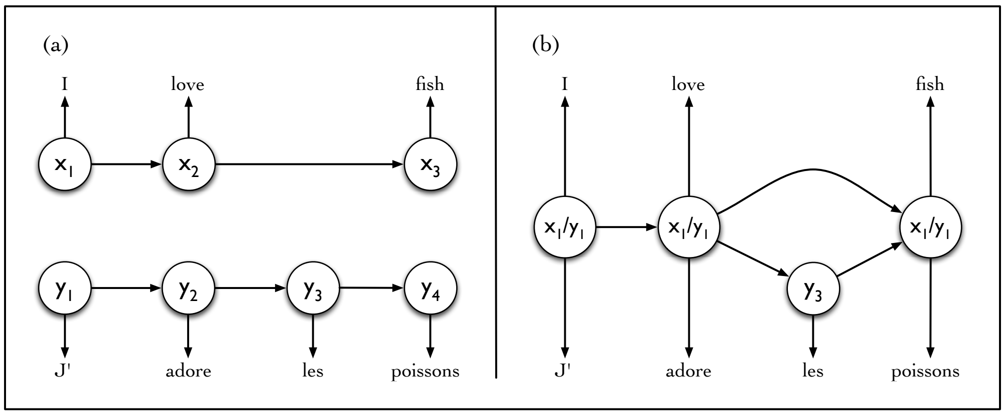


Figure 1: (a) Graphical structure of two standard monolingual HMM's. (b) Graphical structure of our bilingual model based on word alignments.

language barrier, while respecting language-specific idiosyncrasies such as tag inventory, selection, and order. We assume that for pairs of words that share similar semantic or syntactic function, the associated tags will be statistically correlated, though not necessarily identical. We use such word pairs as the bilingual anchors of our model, allowing cross-lingual information to be shared via joint tagging decisions. We use standard tools from machine translation to identify these aligned words, and thereafter our model treats them as fixed and observed data. To avoid cycles, we remove crossing edges from the alignments.

For unaligned parts of the sentence, the tag and word selections are identical to standard monolingual HMM's. Figure 1 shows an example of the bilingual graphical structure we use, in comparison to two independent monolingual HMM's.

We formulate a hierarchical Bayesian model that exploits both language-specific and cross-lingual patterns to explain the observed bilingual sentences. We present a generative story in which the observed words are produced by the hidden tags and model parameters. In Section 4, we describe how to infer the posterior distribution over these hidden variables, given the observations.

3.1 Generative Model

Our generative model assumes the existence of two tagsets, T and T', and two vocabularies W and W', one of each for each language. For ease of exposition, we formulate our model with bigram tag dependencies. However, in our experiments we used a trigram model, which is a trivial extension of the model discussed here and in the next section.

1. For each tag $t \in T$, draw a *transition* distribution ϕ_t over tags T, and an *emission* distribution θ_t over words W, both from symmetric Dirichlet priors.[1]

2. For each tag $t \in T'$, draw a *transition* distribution ϕ'_t over tags T', and an *emission* distribution θ'_t over words W', both from symmetric Dirichlet priors.

3. Draw a bilingual *coupling* distribution ω over tag pairs $T \times T'$ from a symmetric Dirichlet prior.

4. For each bilingual parallel sentence:

   (a) Draw an alignment a from an alignment distribution A (see the following paragraph for formal definitions of a and A),

   (b) Draw a bilingual sequence of part-of-speech tags $(x_1, ..., x_m)$, $(y_1, ..., y_n)$ according to:
   $P(x_1, ..., x_m,\ \ y_1, ..., y_n | a, \phi, \phi', \omega)$. [2]
   This joint distribution is given in equation 1.

[1]The Dirichlet is a probability distribution over the simplex, and is conjugate to the multinomial (Gelman et al., 2004).

[2]Note that we use a special end state rather than explicitly modeling sentence length. Thus the values of m and n depend on the draw.

(c) For each part-of-speech tag x_i in the first language, emit a word from W: $e_i \sim \theta_{x_i}$,

(d) For each part-of-speech tag y_j in the second language, emit a word from W': $f_j \sim \theta'_{y_j}$.

We define an alignment a to be a set of one-to-one integer pairs with no crossing edges. Intuitively, each pair $(i, j) \in a$ indicates that the words e_i and f_j share some common role in the bilingual parallel sentences. In our experiments, we assume that alignments are directly observed and we hold them fixed. From the perspective of our generative model, we treat alignments as drawn from a distribution A, about which we remain largely agnostic. We only require that A assign zero probability to alignments which either: *(*i) align a single index in one language to multiple indices in the other language or *(*ii) contain crossing edges. The resulting alignments are thus one-to-one, contain no crossing edges, and may be sparse or even possibly empty. Our technique for obtaining alignments that display these properties is described in Section 5.

Given an alignment a and sets of transition parameters ϕ and ϕ', we factor the conditional probability of a bilingual tag sequence $(x_1, ...x_m),\ (y_1, ..., y_n)$ into transition probabilities for unaligned tags, and joint probabilities over aligned tag pairs:

$$P(x_1, ..., x_m,\ y_1, ..., y_n | a, \phi, \phi', \omega) = \prod_{\text{unaligned } i} \phi_{x_{i-1}}(x_i) \cdot \prod_{\text{unaligned } j} \phi'_{y_{j-1}}(y_j) \cdot \prod_{(i,j) \in a} P(x_i, y_j | x_{i-1}, y_{j-1}, \phi, \phi', \omega) \quad (1)$$

Because the alignment contains no crossing edges, we can model the tags as generated sequentially by a stochastic process. We define the distribution over aligned tag pairs to be a product of each language's transition probability and the coupling probability:

$$P(x_i, y_j | x_{i-1}, y_{j-1}, \phi, \phi', \omega) = \frac{\phi_{x_{i-1}}(x_i)\ \phi'_{y_{j-1}}(y_j)\ \omega(x_i, y_j)}{Z} \quad (2)$$

The normalization constant here is defined as:

$$Z = \sum_{x,y} \phi_{x_{i-1}}(x)\ \phi'_{y_{j-1}}(y)\ \omega(x, y)$$

This factorization allows the language-specific transition probabilities to be shared across aligned and unaligned tags. In the latter case, the addition of the coupling parameter ω gives the tag pair an additional role: that of multilingual anchor. In essence, the probability of the aligned tag pair is a product of three experts: the two transition parameters and the coupling parameter. Thus, the combination of a high probability transition in one language and a high probability coupling can resolve cases of inherent transition uncertainty in the other language. In addition, any one of the three parameters can "veto" a tag pair to which it assigns low probability.

To perform inference in this model, we predict the bilingual tag sequences with maximal probability given the observed words and alignments, while integrating over the transition, emission, and coupling parameters. To do so, we use a combination of sampling-based techniques.

4 Inference

The core element of our inference procedure is Gibbs sampling (Geman and Geman, 1984). Gibbs sampling begins by randomly initializing all unobserved random variables; at each iteration, each random variable z_i is sampled from the conditional distribution $P(z_i|\mathbf{z}_{-i})$, where $\mathbf{z}_{-i}$ refers to all variables other than z_i. Eventually, the distribution over samples drawn from this process will converge to the unconditional joint distribution $P(\mathbf{z})$ of the unobserved variables. When possible, we avoid explicitly sampling variables which are not of direct interest, but rather integrate over them—this technique is known as "collapsed sampling," and can reduce variance (Liu, 1994).

We sample: *(i)* the bilingual tag sequences $(\mathbf{x}, \mathbf{y})$, *(ii)* the two sets of transition parameters ϕ and ϕ', and *(iii)* the coupling parameter ω. We integrate over the emission parameters θ and θ', whose priors are Dirichlet distributions with hyperparameters θ_0 and θ'_0. The resulting emission distribution over words e_i, given the other words $\mathbf{e}_{-i}$, the tag sequences $\mathbf{x}$

and the emission prior θ_0, can easily be derived as:

$$\begin{aligned} P(e_i|\mathbf{x}, \mathbf{e}_{-i}, \theta_0) &= \int_{\theta_{x_i}} \theta_{x_i}(e_i)\, P(\theta_{x_i}|\theta_0)\; d\theta_{x_i} \\ &= \frac{n(x_i, e_i) + \theta_0}{n(x_i) + W_{x_i}\theta_0} \end{aligned} \tag{3}$$

Here, $n(x_i)$ is the number of occurrences of the tag x_i in $\mathbf{x}_{-i}$, $n(x_i, e_i)$ is the number of occurrences of the tag-word pair (x_i, e_i) in $(\mathbf{x}_{-i}, \mathbf{e}_{-i})$, and W_{x_i} is the number of word types in the vocabulary W that can take tag x_i. The integral is tractable due to Dirichlet-multinomial conjugacy (Gelman et al., 2004).

We will now discuss, in turn, each of the variables that we sample. Note that in all cases we condition on the other sampled variables as well as the observed words and alignments, $\mathbf{e}$, $\mathbf{f}$ and a, which are kept fixed throughout.

4.1 Sampling Part-of-speech Tags

This section presents the conditional distributions that we sample from to obtain the part-of-speech tags. Depending on the alignment, there are several scenarios. In the simplest case, both the tag to be sampled and its succeeding tag are not aligned to any tag in the other language. If so, the sampling distribution is identical to the monolingual case, including only terms for the emission (defined in equation 3), and the preceding and succeeding transitions:

$$\begin{aligned} &P(x_i|\mathbf{x}_{-i}, \mathbf{y}, \mathbf{e}, \mathbf{f}, a, \phi, \phi', \omega, \theta_0, \theta_0') \propto \\ &\qquad P(e_i|\mathbf{x}, \mathbf{e}_{-i}, \theta_0)\; \phi_{x_{i-1}}(x_i)\; \phi_{x_i}(x_{i+1}). \end{aligned}$$

For an aligned tag pair (x_i, y_j), we sample the identity of the tags jointly. By applying the chain rule we obtain terms for the emissions in both languages and a joint term for the transition probabilities:

$$\begin{aligned} &P(x_i, y_j|\mathbf{x}_{-i}, \mathbf{y}_{-j}, \mathbf{e}, \mathbf{f}, a, \phi, \phi', \omega, \theta_0, \theta_0') \propto \\ &\qquad P(e_i|\mathbf{x}, \mathbf{e}_{-i}, \theta_0) P(f_j|\mathbf{y}, \mathbf{f}_{-j}, \theta_0') \\ &\qquad\qquad P(x_i, y_j|\mathbf{x}_{-i}, \mathbf{y}_{-j}, a, \phi, \phi', \omega) \end{aligned}$$

The expansion of the joint term depends on the alignment of the succeeding tags. In the case that the successors are not aligned, we have a product of the bilingual coupling probability and four transition probabilities (preceding and succeeding transitions in each language):

$$\begin{aligned} &P(x_i, y_j|\mathbf{x}_{-i}, \mathbf{y}_{-j}, a, \phi, \phi', \omega) \propto \\ &\quad \omega(x_i, y_j)\phi_{x_{i-1}}(x_i)\; \phi'_{y_{j-1}}(y_j)\; \phi_{x_i}(x_{i+1})\; \phi'_{y_j}(y_{j+1}) \end{aligned}$$

Whenever one or more of the succeeding tags is aligned, the sampling formulas must account for the effect of the sampled tag on the joint probability of the succeeding tags, which is no longer a simple multinomial transition probability. We give the formula for one such case—when we are sampling an aligned tag pair (x_i, y_j), whose succeeding tags (x_{i+1}, y_{j+1}) are also aligned to one another:

$$\begin{aligned} &P(x_i, y_j|\mathbf{x}_{-i}, \mathbf{y}_{-j}, a, \phi, \phi', \omega) \propto \omega(x_i, y_j) \\ &\quad \cdot \phi_{x_{i-1}}(x_i)\, \phi'_{y_{j-1}}(y_j) \left[\frac{\phi_{x_i}(x_{i+1})\, \phi'_{y_j}(y_{j+1})}{\sum_{x,y} \phi_{x_i}(x)\, \phi'_{y_j}(y)\, \omega(x, y)} \right] \end{aligned}$$

Similar equations can be derived for cases where the succeeding tags are not aligned to each other, but to other tags.

4.2 Sampling Transition Parameters and the Coupling Parameter

When computing the joint probability of an aligned tag pair (Equation 2), we employ the transition parameters ϕ, ϕ' and the coupling parameter ω in a normalized product. Because of this, we can no longer regard these parameters as simple multinomials, and thus can no longer sample them using the standard closed formulas.

Instead, to resample these parameters, we resort to the Metropolis-Hastings algorithm as a subroutine within Gibbs sampling (Hastings, 1970). Metropolis-Hastings is a Markov chain sampling technique that can be used when it is impossible to directly sample from the posterior. Instead, samples are drawn from a *proposal* distribution and then stochastically accepted or rejected on the basis of: their likelihood, their probability under the proposal distribution, and the likelihood and proposal probability of the previous sample.

We use a form of Metropolis-Hastings known as an *independent sampler*. In this setup, the proposal distribution does not depend on the value of the previous sample, although the accept/reject decision does depend on the previous model likelihood. More formally, if we denote the proposal distribution as $Q(z)$, the target distribution as $P(z)$, and the previous sample as z, then the probability of accepting a new sample $z^* \sim Q$ is set at:

$$\min\left\{1, \frac{P(z^*)\,Q(z)}{P(z)\,Q(z^*)}\right\}$$

Theoretically any non-degenerate proposal distribution may be used. However, a higher acceptance rate and faster convergence is achieved when the proposal Q is a close approximation of P. For a particular transition parameter ϕ_x, we define our proposal distribution Q to be Dirichlet with parameters set to the bigram counts of the tags following x in the sampled tag data. Thus, the proposal distribution for ϕ_x has a mean proportional to these counts, and is thus likely to be a good approximation to the target distribution.

Likewise for the coupling parameter ω, we define a Dirichlet proposal distribution. This Dirichlet is parameterized by the counts of aligned tag pairs (x, y) in the current set of tag samples. Since this sets the mean of the proposal to be proportional to these counts, this too is likely to be a good approximation to the target distribution.

4.3 Hyperparameter Re-estimation

After every iteration of Gibbs sampling the hyperparameters θ_0 and θ'_0 are re-estimated using a single Metropolis-Hastings move. The proposal distribution is set to a Gaussian with mean at the current value and variance equal to one tenth of the mean.

5 Experimental Set-Up

Our evaluation framework follows the standard procedures established for unsupervised part-of-speech tagging. Given a tag dictionary (i.e., a set of possible tags for each word type), the model has to select the appropriate tag for each token occurring in a text. We also evaluate tagger performance when only incomplete dictionaries are available (Smith and Eisner, 2005; Goldwater and Griffiths, 2007). In both scenarios, the model is trained only using untagged text.

In this section, we first describe the parallel data and part-of-speech annotations used for system evaluation. Next we describe a monolingual baseline and our procedures for initialization and hyperparameter setting.

Data As a source of parallel data, we use Orwell's novel "Nineteen Eighty Four" in the original English as well as translations to three Slavic languages — Bulgarian, Serbian and Slovene. This data is distributed as part of the Multext-East corpus which is publicly available. The corpus provides detailed morphological annotation at the world level, including part-of-speech tags. In addition a lexicon for each language is provided.

We obtain six parallel corpora by considering all pairings of the four languages. We compute word level alignments for each language pair using Giza++. To generate one-to-one alignments at the word level, we intersect the one-to-many alignments going in each direction and automatically remove crossing edges in the order in which they appear left to right. This process results in alignment of about half the tokens in each bilingual parallel corpus. We treat the alignments as fixed and observed variables throughout the training procedure.

The corpus consists of 94,725 English words (see Table 2). For every language, a random three quarters of the data are used for learning the model while the remaining quarter is used for testing. In the test set, only monolingual information is made available to the model, in order to simulate future performance on non-parallel data.

	Tokens	Tags/Token
SR	89,051	1.41
SL	91,724	1.40
BG	80,757	1.34
EN	94,725	2.58

Table 2: Corpus statistics: SR=Serbian, SL=Slovene, EN=English, BG=Bulgarian

Tagset The Multext-East corpus is manually annotated with detailed morphosyntactic information. In our experiments, we focus on the main syntactic category encoded as a first letter of the labels. The annotation distinguishes between 13 parts-of-speech, of which 11 are common for all languages

	Random	Monolingual Unsupervised	Monolingual Supervised	Trigram Entropy
EN	56.24	90.71	96.97	1.558
BG	82.68	88.88	96.96	1.708
SL	84.70	87.41	97.31	1.703
SR	83.41	85.05	96.72	1.789

Table 1: Monolingual tagging accuracy for English, Bulgarian, Slovene, and Serbian for two unsupervised baselines (random tag selection and a Bayesian HMM (Goldwater and Griffiths, 2007)) as well as a supervised HMM. In addition, the trigram part-of-speech tag entropy is given for each language.

in our experiments.[3]

In the Multext-East corpus, punctuation marks are not annotated. We expand the tag repository by defining a separate tag for all punctuation marks. This allows the model to make use of any transition or coupling patterns involving punctuation marks. We do not consider punctuation tokens when computing model accuracy.

Table 2 shows the tag/token ratio for these languages. For Slavic languages, we use the tag dictionaries provided with the corpus. For English, we use a different process for dictionary construction. Using the original dictionary would result in the tag/token ratio of 1.5, in comparison to the ratio of 2.3 observed in the Wall Street Journal (WSJ) corpus. To make our results on English tagging more comparable to previous benchmarks, we expand the original dictionary of English tags by merging it with the tags from the WSJ dictionary. This process results in a tag/token ratio of 2.58, yielding a slightly more ambiguous dictionary than the one used in previous tagging work. [4]

Monolingual Baseline As our monolingual baseline we use the unsupervised Bayesian HMM model of Goldwater and Griffiths (2007) (BHMM1). This model modifies the standard HMM by adding priors and by performing Bayesian inference. Its is in line with state-of-the-art unsupervised models. This model is a particulary informative baseline, since our model reduces to this baseline model when there are no alignments in the data. This implies that any performance gain over the baseline can only be attributed to the multilingual aspect of our model. We used our own implementation after verifying that its performance on WSJ was identical to that reported in (Goldwater and Griffiths, 2007).

[3]The remaining two tags are Particle and Determiner; The English tagset does not include *Particle* while the other three languages Serbian, Slovene and Bulgarian do not have *Determiner* in their tagset.

[4]We couldn't perform the same dictionary expansion for the Slavic languages due to a lack of additional annotated resources.

Supervised Performance In order to provide a point of comparison, we also provide supervised results when an annotated corpus is provided. We use the standard supervised HMM with Viterbi decoding.

Training and Testing Framework Initially, all words are assigned tags randomly from their tag dictionaries. During each iteration of the sampler, aligned tag pairs and unaligned tags are sampled from their respective distributions given in Section 4.1 above. The hyperparameters θ_0 and θ_0' are initialized with the values learned during monolingual training. They are re-estimated after every iteration of the sampler using the Metropolis Hastings algorithm. The parameters ϕ and ϕ' are initially set to trigram counts and the ω parameter is set to tag pair counts of aligned pairs. After every 40 iterations of the sampler, a Metropolis Hastings subroutine is invoked that re-estimates these parameters based on the current counts. Overall, the algorithm is run for 1000 iterations of tag sampling, by which time the resulting log-likelihood converges to stable values. Each Metropolis Hastings subroutine samples 20 values, with an acceptance ratio of around 1/6, in line with the standard recommended values.

After training, trigram and word emission probabilities are computed based on the counts of tags assigned in the final iteration. For smoothing, the final sampled values of the hyperparameters are used. The highest probability tag sequences for each monolingual test set are then predicted using trigram Viterbi decoding. We report results averaged over five complete runs of all experiments.

6 Results

Complete Tag Dictionary In our first experiment, we assume that a complete dictionary listing the possible tags for every word is provided in each language. Table 1 shows the monolingual results of a random baseline, an unsupervised Bayesian HMM and a supervised HMM. Table 3 show the results of our bilingual models for different language pairings while repeating the monolingual unsupervised results from Table 1 for easy comparison. The final column indicates the absolute gain in performance over this monolingual baseline.

Across all language pairs, the bilingual model consistently outperforms the monolingual baseline. All the improvements are statistically significant by a Fisher sign test at $p < 0.05$. For some language pairs, the gains are quite high. For instance, the pairing of Serbian and Slovene (two closely related languages) yields absolute improvements of 6.7 and 7.7 percentage points, corresponding to relative reductions in error of 51.4% and 53.2%. Pairing Bulgarian and English (two distantly related languages) also yields large gains: 5.6 and 1.3 percentage points, corresponding to relative reductions in error of 50% and 14%, respectively.[5]

When we compare the best bilingual result for each language (Table 3, in bold) to the monolingual supervised results (Table 1), we find that for all languages the gap between supervised and unsupervised learning is reduced significantly. For English, this gap is reduced by 21%. For the Slavic languages, the supervised-unsupervised gap is reduced by even larger amounts: 57%, 69%, and 78% for Serbian, Bulgarian, and Slovene respectively.

While all the languages benefit from the bilingual learning framework, some language combinations are more effective than others. Slovene, for instance, achieves a large improvement when paired with Serbian ($+7.7$), a closely related Slavic language, but only a minor improvement when coupled with English ($+1.3$). On the other hand, for Bulgarian, the best performance is achieved when coupling with English ($+5.6$) rather than with closely related Slavic languages ($+3.1$ and $+2.4$). As these results show, an optimal pairing cannot be predicted based solely on the family connection of paired languages.

To gain a better understanding of this variation in performance, we measured the internal tag entropy of each language as well as the cross-lingual tag entropy of language pairs. For the first measure, we computed the conditional entropy of a tag decision given the previous two tags. Intuitively, this should correspond to the inherent structural uncertainty of part-of-speech decisions in a language. In fact, as Table 1 shows, the trigram entropy is a good indicator of the relative performance of the monolingual baseline. To measure the cross-lingual tag entropies of language pairs, we considered all bilingual aligned tag pairs, and computed the conditional entropy of the tags in one language given the tags in the other language. This measure should indicate the amount of information that one language in a pair can provide the other. The results of this anal-

	Entropy	Mono-lingual	Bilingual	Absolute Gain
EN	0.566	90.71	91.01	+0.30
SR	0.554	85.05	90.06	+5.03
EN	0.578	90.71	92.00	+1.29
BG	0.543	88.88	**94.48**	**+5.61**
EN	0.571	90.71	**92.01**	**+1.30**
SL	0.568	87.41	88.54	+1.13
SL	0.494	87.41	**95.10**	**+7.69**
SR	0.478	85.05	**91.75**	**+6.70**
BG	0.568	88.88	91.95	+3.08
SR	0.588	85.05	86.58	+1.53
BG	0.579	88.88	90.91	+2.04
SL	0.609	87.41	88.20	+0.79

Table 3: The tagging accuracy of our bilingual models on different language pairs, when a full tag dictionary is provided. The Monolingual Unsupervised results from Table 1 are repeated for easy comparison. The first column shows the cross-lingual entropy of a tag when the tag of the aligned word in the other language is known. The final column shows the absolute improvement over the monolingual Bayesian HMM. The best result for each language is shown in boldface.

[5]The accuracy of the monolingual English tagger is relatively high compared to the 87% reported by (Goldwater and Griffiths, 2007) on the WSJ corpus. We attribute this discrepancy to the slight differences in tag inventory used in our dataset. For example, when *Particles* and *Prepositions* are merged in the WSJ corpus (as they happen to be in our tag inventory and corpus), the performance of Goldwater's model on WSJ is similar to what we report here.

	Mono-lingual	Bilingual	Absolute Gain
EN	63.57	68.22	+4.66
SR	41.14	54.73	+13.59
EN	63.57	**71.34**	**+7.78**
BG	53.19	**62.55**	**+9.37**
EN	63.57	66.48	+2.91
SL	49.90	53.77	+3.88
SL	49.90	**59.68**	**+9.78**
SR	41.14	54.08	+12.94
BG	53.19	54.22	+1.04
SR	41.14	**56.91**	**+15.77**
BG	53.19	55.88	+2.70
SL	49.90	58.50	+8.60

Table 4: Tagging accuracy for Bilingual models with reduced dictionary: Lexicon entries are available for only the 100 most frequent words, while all other words become fully ambiguous. The improvement over the monolingual Bayesian HMM trained under similar circumstances is shown. The best result for each language is shown in boldface.

ysis are given in the first column of Table 3. We observe that the cross-lingual entropy is lowest for the Serbian and Slovene pair, corresponding with their large gain in performance. Bulgarian, on the other hand, has lowest cross-lingual entropy when paired with English. This corresponds with the fact that English provides Bulgarian with its largest performance gain. In general, we find that the largest performance gain for any language is achieved when minimizing its cross-lingual entropy.

Reduced Tag Dictionary We also conducted experiments to investigate the impact of the dictionary size on the performance of the bilingual model. Here, we provide results for the realistic scenario where only a very small dictionary is present. Table 4 shows the performance when a tag dictionary for the 100 most frequent words is present in each language. The bilingual model's results are consistently and significantly better than the monolingual baseline for all language pairs.

7 Conclusion

We have demonstrated the effectiveness of multilingual learning for unsupervised part-of-speech tagging. The key hypothesis of multilingual learning is that by combining cues from multiple languages, the structure of each becomes more apparent. We formulated a hierarchical Bayesian model for jointly predicting bilingual streams of tags. The model learns language-specific features while capturing cross-lingual patterns in tag distribution. Our evaluation shows significant performance gains over a state-of-the-art monolingual baseline.

Acknowledgments

The authors acknowledge the support of the National Science Foundation (CAREER grant IIS-0448168 and grant IIS-0835445) and the Microsoft Research Faculty Fellowship. Thanks to Michael Collins, Amir Globerson, Lillian Lee, Yoong Keok Lee, Maria Polinsky and the anonymous reviewers for helpful comments and suggestions. 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

Michele Banko and Robert C. Moore. 2004. Part-of-speech tagging in context. In *Proceedings of the COLING*, pages 556–561.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1991. Word-sense disambiguation using statistical methods. In *Proceedings of the ACL*, pages 264–270.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the ACL*, pages 263–270.

Ido Dagan, Alon Itai, and Ulrike Schwall. 1991. Two languages are more informative than one. In *Proceedings of the ACL*, pages 130–137.

Mona Diab and Philip Resnik. 2002. An unsupervised method for word sense tagging using parallel corpora. In *Proceedings of the ACL*, pages 255–262.

Anna Feldman, Jirka Hana, and Chris Brew. 2006. A cross-language approach to rapid creation of new morpho-syntactically annotated resources. In *Proceedings of LREC*, pages 549–554.

Andrew Gelman, John B. Carlin, Hal .S. Stern, and Donald .B. Rubin. 2004. *Bayesian data analysis*. Chapman and Hall/CRC.

S. Geman and D. Geman. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.

Dmitriy Genzel. 2005. Inducing a multilingual dictionary from a parallel multitext in related languages. In *Proceedings of HLT/EMNLP*, pages 875–882.

Sharon Goldwater and Thomas L. Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the ACL*, pages 744–751.

Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. *Proceedings of HLT-NAACL*, pages 320–327.

W. K. Hastings. 1970. Monte carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109.

Mark Johnson. 2007. Why doesn't EM find good HMM POS-taggers? In *Proceedings of EMNLP/CoNLL*, pages 296–305.

Jun S. Liu. 1994. The collapsed Gibbs sampler in Bayesian computations with applications to a gene regulation problem. *Journal of the American Statistical Association*, 89(427):958–966.

Bernard Merialdo. 1994. Tagging english text with a probabilistic model. *Computational Linguistics*, 20(2):155–171.

Hwee Tou Ng, Bin Wang, and Yee Seng Chan. 2003. Exploiting parallel texts for word sense disambiguation: an empirical study. In *Proceedings of the ACL*, pages 455–462.

Sebastian Padó and Mirella Lapata. 2006. Optimal constituent alignment with edge covers for semantic projection. In *Proceedings of ACL*, pages 1161 – 1168.

Philip Resnik and David Yarowsky. 1997. A perspective on word sense disambiguation methods and their evaluation. In *Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?*, pages 79–86.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the ACL*, pages 354–362.

Benjamin Snyder and Regina Barzilay. 2008. Unsupervised multilingual learning for morphological segmentation. In *Proceedings of the ACL/HLT*, pages 737–745.

Kristina Toutanova and Mark Johnson. 2008. A Bayesian lda-based model for semi-supervised part-of-speech tagging. In *Advances in Neural Information Processing Systems 20*, pages 1521–1528. MIT Press.

Dekai Wu and Hongsing Wong. 1998. Machine translation with a stochastic grammatical channel. In *Proceedings of the ACL/COLING*, pages 1408–1415.

Chenhai Xi and Rebecca Hwa. 2005. A backoff model for bootstrapping resources for non-english languages. In *Proceedings of EMNLP*, pages 851 – 858.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual pos taggers and np bracketers via robust projection across aligned corpora. In *Proceedings of the NAACL*, pages 1–8.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2000. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of HLT*, pages 161–168.

Part-of-Speech Tagging for English-Spanish Code-Switched Text

Thamar Solorio and **Yang Liu**
Human Language Technology Research Institute
The University of Texas at Dallas
Richardson, TX 75080, USA
tsolorio,yangl@hlt.utdallas.edu

Abstract

Code-switching is an interesting linguistic phenomenon commonly observed in highly bilingual communities. It consists of mixing languages in the same conversational event. This paper presents results on Part-of-Speech tagging Spanish-English code-switched discourse. We explore different approaches to exploit existing resources for both languages that range from simple heuristics, to language identification, to machine learning. The best results are achieved by training a machine learning algorithm with features that combine the output of an English and a Spanish Part-of-Speech tagger.

1 Introduction

Worldwide the percentage of bilingual speakers is fairly large, and it keeps increasing at a high rate. In the U.S., 18% of the total population speaks a language other than English at home, the majority of which speaks Spanish (U.S. Census Bureau, 2003). A significant percentage of this Spanish-English bilingual population code-switch between the two languages in what is often referred as Spanglish, the mix of Spanish and English. Spanish and English are not the only occurrence of language mixtures. Examples of other popular combinations include Arabic dialects, French and German, Spanish and Catalan, Maltese and English, and English and French. Typically when there are linguistic borders, or when the country has more than one official language, we can find instances of code-switching.

Despite the wide use of code-switched discourse among bilinguals, this linguistic phenomenon has received little attention in the fields of Natural Language Processing and Computational Linguistics. Part-of-Speech (POS) tagging is a well studied problem in these fields. For languages such as English, German, Spanish, and Chinese there are several different POS taggers that reach high accuracies, especially in news text corpora. However, to our knowledge, there is no previous work on developing a POS tagger for text with mixes of languages.

In this paper we present results on the problem of POS tagging English-Spanish code-switched discourse by taking advantage of existing taggers for both languages. This rationale follows the evidence from studies of code-switching on different language pairs, which have shown code-switching to be grammatical according to both languages being switched. We use different heuristics to combine POS tag information from existing monolingual taggers. We also explore the use of different language identification methods to select POS tags from the appropriate monolingual tagger. However, the best results are achieved by a machine learning approach using features generated by the monolingual POS taggers.

The next section presents the facts about code-switching, including some previous work done mainly in linguistics. Then in Section 3 we discuss previous work related to the automated processing of code-switched discourse. In Section 4 we describe the English-Spanish code-switched data set gathered for the experimental evaluation. Section 5 presents the heuristics-based approaches for POS tagging that we explored. In Section 6 we describe our machine learning approach and show

Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, pages 1051–1060,
Honolulu, October 2008. ©2008 Association for Computational Linguistics

results on POS tagging code-switched text. An in depth analysis of results is presented in Section 7, and we conclude this paper with a summary of the findings and directions for future work in Section 8.

2 Rules of Code-switching

In the linguistic, sociolinguistic, psychology, and psycholinguistic literature, bilingualism and the inherent phenomena it exhibits have been studied for nearly a century (Espinosa, 1917; Ervin and Osgood, 1954; Gumperz, 1964; Gumperz and Hernandez-Chavez, 1971; Gumperz, 1971; Sankoff, 1968; Lipski, 1978). Despite the numerous previous studies of linguistic characteristics of bilingualism, there is no clear consensus on the terminology related to language alternation patterns in bilingual speakers. The alternation of languages within a sentence is known as code-mixing, but it has also been referred as intrasentential code-switching, and intrasentential alternation (Poplack, 1980; Grosjean, 1982; Ardila, 2005). Alternation across sentence boundaries is known as intersentential code-switching, or just code-switching. In the rest of this paper we will refer to the mixing of languages as code-switching. When necessary, we will differentiate the type of code-switching by referring to alternations within sentences as intrasentential code-switching and alternations across sentence boundaries as intersentential code-switching.

Linguistic phenomena in bilingual speakers have been analyzed on different language pairs, including English-French, English-Dutch, Finish-English, Arabic-French, and Spanish-English, to name a few. There is a general agreement that code-switched patterns are not generated randomly; according to these studies, they follow specific grammatical rules. Furthermore, some studies suggest that, if these rules are violated, the resulting discourse will sound unnatural (Toribio, 2001b; Toribio, 2001a). The following shows the rules governing code-switching discourse described in several studies (Poplack, 1980; Poplack, 1981; Sankoff, 1981; Sankoff, 1998a).

- Switches can take place only between full word boundaries. This is also known as the free morpheme constraint.
- Monolingual constructs within the sentence will follow the grammatical rules of the monolingual fragment.
- Permissible switch points are those that do not violate the order of adjacent constituents on both sides of the switch point of either of the languages. This is called the equivalence constraint.

Although these rules are somewhat controversial, and most of the studies on this area have been conducted on small samples, we cannot ignore the fact that patterns bearing the above rules have emerged in different bilingual communities with different backgrounds.

3 Automated Processing of Code-Switched Discourse

A previous work related to the processing of code-switched text deals with language identification on English-Maltese code-switched SMS messages (Rosner and Farrugia, 2007). In addition to dealing with intrasentential code-switching, they have to deal with text where misspellings and ad hoc word contractions abound. What Rosner and Farrigua have found to work best for language identification in this noisy domain is a combination of a bigram Hidden Markov Model, trained on language transitions, and a trigram character Markov Model for handling unknown words. In another related work, Franco and Solorio present preliminary results on training a language model for Spanish-English code-switched text (Franco and Solorio, 2007). To evaluate their language model, they asked a human subject to judge sentences generated by a PCFG induced from training data and the language model. However, they only used one human judge.

Regarding the automated POS tagging and parsing of code-mixed utterances there is little prior work. To the best of our knowledge, there is no parser, nor POS tagger, currently available for the syntactic analysis of this type of discourse. There are theoretical approaches that propose formalisms to represent the structure of code-switched utterances and describe a framework for parsing and generating mixed sentences, for example for Marathi and English (Joshi, 1982), or Hindi and English (Goyal *et al.*, 2003). Sankoff proposed a production model of bilingual discourse that accounts for the equivalence constraint and the unpredictability of code-switching (Sankoff, 1998a; Sankoff, 1998b). His real-time production model draws on the alternation of fragments from two virtual monolingual sentences. It also accounts for other types of code-switching such as repetition-translation and insertional code-switching. But no statistical assessment has been conducted on real corpora.

Our goal is to develop a POS tagger for code-switched utterances, which is the first step of the syntactic analysis of any language. Among the challenges we face is the lack of a representative sample of code-mixed discourse. Most POS taggers are built using large collections, usually at least a million words, such as the Brown corpus (Kucera and Francis, 1967), the Wall Street Journal corpus (Paul and Baker, 1992), or the Switchboard corpus (Godfrey *et al.*, 1992). Currently, there is no annotation of code-switched text of comparable size. But in contrast to the lack of linguistic resources available for Spanish-English code-mixed discourse, English and Spanish have sufficient resources, especially English. Thus, rather than starting from scratch, we will draw on existing taggers for both languages, which will reduce the amount of code-switched data needed. Some examples of POS taggers that perform reasonably well on monolingual text of each language can be found in (Brants, 2000; Brill, 1992; Carreras and Padró, 2002; Charniak, 1993; Ratnaparkhi, 1996; Schmid, 1994). However, these tools are designed to work on monolingual text, therefore if applied as they are to code-switched text, their accuracy will decrease by a large margin. In the following sections we will explore different methods for combining monolingual taggers.

4 Data Set

Data collections that have instances of Spanish-English code-switching, Spanglish for short, are not easily found since code-switching is primarily used in spoken form. To gather data we recorded a conversation among three staff members of a southwest university in the U.S. The three speakers come from a highly bilingual background and code-switch regularly when speaking among themselves, or other bilingual speakers.

This recording session has around 39 minutes of continuous speech (922 sentences, about 8k words) and was transcribed and annotated with POS tags by a human annotator. The annotations were later revised by a different annotator but no inter-annotator agreement was measured. The POS tag set used in the annotation is the combination of the tag sets from the English and the Spanish Tree Taggers (see Section 5). The vocabulary of the transcription has a total of 1,516 different word forms[1]. In the conversation a total of 239 switches were identified manually, out of which 129 are intrasentential code-switches, and the rest are intersentential. English is the predominant language used, with a total of 6,020 tokens and 576 monolingual sentences. In contrast, the transcription has close to 2k tokens in Spanish. Table 1 shows examples of code-switching taken from the recorded conversation; (a) and (b) are instances of intrasentential code-switching, and (c) shows intersentential code-switching.

Table 1: Excerpts taken from the Spanglish data set.

Spanglish	**English Translation**
(a)*Entonces le dió el virus y no se lo atendió and the virus spread through his body.*	(a)Then he got the virus and he didn't receive treatment and the virus spread through his body.
(b)*Cuando yo lo vi he looked pretty bad.*	(b)When I saw him he looked pretty bad.
(c)*I think she was taller than he was.*	(c)I think she was taller than he was.
Y un carácter muy bonito también ella.	And a very nice character she as well.
Very easy going.	Very easy going.

5 Rule-based Methods for Exploiting Existing Resources

In this section we present several heuristics-based methods for POS tagging code-switched text. First, we describe the monolingual taggers used in this work. Then we present the different approaches explored and contrast their performance.

[1]This transcription and the audio file are freely available for research purposes by contacting the first author.

5.1 Monolingual Taggers

We used the Tree Tagger (Schmid, 1994) for this work because of the following considerations:

1. It has both English and Spanish versions. The English tagger uses a slightly modified version of the Penn Treebank tag set and was trained and evaluated on different portions of the Penn Treebank, reaching a POS tagging accuracy of 96.36%. The Spanish one uses a different tagset with 75 different POS tags[2] and was trained on the Spanish CRATER corpus.
2. The transition probabilities are estimated using a modified version of the ID3 decision tree algorithm (Quinlan, 1986), which provides more freedom to learn contextual cues than n-grams.
3. Both taggers include a special tag for foreign words, PE for Spanish and FW for English. We do not expect this tag to identify correctly all foreign words, but when available this information will be exploited.
4. The Tree tagger generates probability estimates on the tags that can be used as features.
5. Finally, when the tagger fails to lemmatize a word it outputs the special token $\langle$*unknown*$\rangle$. This information can be used as a hint of words that do not belong to that particular language.

5.2 Heuristic-based Systems

For all heuristics the complete Spanglish data set was given to both taggers as a single text, then the final tag for each word was selected from the output of the taggers according to the different heuristics. Table 2 shows the tagging accuracies of the different heuristics we explored, which are explained below.

1. Using the monolingual tagger. Here we simply give the Spanglish text to the Spanish and the English tree tagger. We expect from both taggers a performance degradation due to the inclusion of foreign words in code-switching, as compared against their accuracy on monolingual texts. Another complicating factor to keep in mind is that we are dealing with spoken language. Hesitations, fillers, disfluencies, and interruption points, such as *Umm*, *Mmmhmm*, and *Uh-huh*, are frequently observed in speech and it is well known that they complicate the POS tagging task. The tagging accuracy from using the individual taggers is rather low, 26% for the Spanish tagger and 54% for the English one. The large difference between the two taggers can be attributed to the fact that the majority of the words in the corpus are in English.

[2]The authors were unable to identify the source of the Spanish tagset.

Table 2: Accuracy on POS tagging Spanglish text using simple heuristics for combining the output of the English and Spanish tagger.

	Heuristic	**Accuracy (%)**
1	Spanish Tree Tagger	25.99
	English Tree Tagger	54.59
2	Highest prob tag or English	51.51
	Highest prob tag or Spanish	49.16
3	Prob + special tags + lemmas	64.27
4	Dictionary-based Language Id	**86.03**
	Character 5-grams Language Id	81.46
	Human Language Id	89.72

2. Using confidence thresholds. The Tree Tagger can output probabilities for each tag, showing the confidence of the tagger on each particular tag. To use this information we choose for each word the tag from the tagger with the highest confidence. When there is a tie we use either the English or the Spanish tag. Table 2 shows the results for the two cases. The "Highest prob tag or English" heuristic gives an accuracy of 51%, which is almost as accurate as using only the English tagger. The "Highest prob tag or Spanish" achieves an accuracy of 49%, which is an improvement over using only the monolingual Spanish tagger, but it is still below the accuracy of the English monolingual tagger. This is also possibly due to the task being easier for the English tagger.

3. Combining confidence thresholds with knowledge from special tags and lemmas. This heuristic uses confidence thresholds combined with decisions based on the special tags, described in Section 5.1, and the unknown lemmas found. Let $POS_E(w_i)$ and $POS_S(w_i)$ be the POS tags assigned to word w_i by the English and Spanish tagger respectively; and let $Prob_E(w_i)$ and $Prob_S(w_i)$ be the confidence scores of POS tags for word w_i computed by the English and Spanish tree taggers, respectively. For each word w_i in the text, the final POS tag,

$POS_F(w_i)$, will be assigned as follows:

1. **If** $POS_E(w_i) = FW$, **then** $POS_F(w_i) \leftarrow POS_S(w_i)$
2. **Else if** $POS_S(w_i) = PE$, **then** $POS_F(w_i) \leftarrow POS_E(w_i)$
3. **Else if** $POS_E(w_i) = \langle unknown \rangle$, **then** $POS_F(w_i) \leftarrow POS_S(w_i)$
4. **Else if** $POS_S(w_i) = \langle unknown \rangle$, **then** $POS_F(w_i) \leftarrow POS_E(w_i)$
5. **Else if** $Prob_E(w_i) > Prob_S(w_i)$, **then** $POS_F(w_i) \leftarrow POS_E(w_i)$
6. **Else** $POS_F(w_i) \leftarrow POS_S(w_i)$

This heuristic performs better than the other methods explored so far, yielding an accuracy of 64.27%. It seems that knowledge of the taggers can be used to improve results. However, POS tagging accuracy is still poor.
4. Selecting POS tags based on automated language identification. We used two different strategies for automatically identifying the language at the word level. One is based on dictionary look-up and the other is character-based language models. For the first approach, every word in the text is searched in the English and Spanish dictionaries. If a word is found in the English dictionary, then we identify that word as belonging to English and the POS tags from the English tagger are used for that word and the following ones, until a word is found in the Spanish dictionary. Similarly, for a word not found in the English dictionary, but found in the Spanish dictionary, we use the Spanish tags until an English word is found. Note that this simple heuristic will always label words that belong to both languages as English, which is also the case for words not found in either dictionary. This dictionary-based method has a language identification accuracy of 94% on the Spanglish corpus.

The character language models were trained on the Agence France Presse (AFP) portions of the Gigaword for English and Spanish, respectively. For each of the words in the Spanglish corpus, we first decide its language by choosing the one with the lowest perplexity, calculated using character n-gram language models, then we use the corresponding POS tag. We experimented with different language model orders, with n ranging from 2 to 6, and found that we achieve the highest accuracy, 81.46%, on POS tagging using a 5-gram language model. This 5-gram method reached a language identification accuracy of 85% for the Spanglish corpus. However, the language identification method using dictionary look-up achieved the best POS tagging result so far: 86.03%. The Spanglish conversation is dominated by every-day language that is easily found in dictionaries, while the text used to train the character based n-gram language models includes vocabulary that is not commonly used in conversations. This can explain why the simple dictionary look-up approach yielded better results for our corpus. Performing manual identification of the language and sending to the appropriate tagger just the corresponding fragments yields a very high POS tagging accuracy, 89.72%. This shows that it is important to deal with the language switches for boosting accuracy. However relying on human annotated language tags would be expensive and for some tasks unfeasible.

6 Machine Learning for POS Tagging Code-Switched Discourse

From Table 2 we can see that, with the exception of the language identification heuristic, accuracies are low for the previous experiments. However, we believe that we can improve results further by using Machine Learning (ML) algorithms trained specifically for this task. In this section we describe the ML setting and present a comparison of the different algorithms we tested.

6.1 Approach

The key point is that the features selected for describing the learning instances are the output from the English and the Spanish taggers. This scheme is similar to a stacked classifier approach (Wolpert, 1992), where the final classifier takes as input the predictions made by the different learners on the first pass and is trained to select the right tag from them, or a different one if the right answer is not available.

The gold-standard POS tags are used as the class label, and instances in this learning task are described by the following attributes:

1. The word (word)
2. English POS tag (E_t)
3. English POS tagger lemma (E_l)
4. English POS tagger confidence (E_p)
5. Spanish POS tag (S_t)
6. Spanish POS tagger lemma (S_l)
7. Spanish POS tagger confidence (S_p)

Feature 1 is just the lexical word form as it appears in the transcript. Features 2 to 4 are generated by the English Tree tagger, while features 5 to 7 are generated by the Spanish Tree tagger. Thus all features are automatically extracted.

6.2 Results

We evaluated experimentally the idea of using ML with different learning algorithms in WEKA (Witten and Frank, 1999). We selected some of the most widely known algorithms, including Support Vector Machines (SVM) with a polynomial kernel of exponent one (Schölkopf and Smola, 2002), Weka's modified version of Quinlan's C4.5 (J48) (Quinlan, 1986), Additive Logistic Regression with Decision Stumps (Logit Boost) (Friedman *et al.*, 1998) and Naive Bayes. The only parameter we modified was for J48 –we enabled the option for reducing error pruning.

Table 3: POS tagging accuracy of Spanglish text with different Machine Learning algorithms. Oracle shows the accuracy achieved when always selecting the right POS tag from the output of both Tree Taggers. Language Id shows accuracy of identifying the language and then choosing the output of the corresponding tagger.

ML Algorithms	Mean Accuracy (%)	Variance
Naive Bayes	88.50	1.9280
SVM	**93.48**	1.2784
Logit Boost	**93.19**	1.4437
J48	91.11	2.1527
Oracle	90.31	-
Language Id	85.80	-

Table 3 shows the average accuracy of 10-fold cross-validation for each classifier together with the variance. SVM and Logit Boost performed the best and the difference between the two algorithms is not significant according to the paired t-test (P-value = 0.1). For comparison, we show the accuracy of the language identification approach together with the oracle accuracy. The oracle is the accuracy achieved when always selecting the right POS tag, when it is available, from the output of both Tree Taggers. We did not expect the oracle's accuracy to be an upper bound on the accuracy for the ML learning algorithm. Our intuition is that the ML algorithm can be trained to identify when the taggers have made a mistake and what the right answer should be. As the results show, the ML approach can indeed outperform the oracle, and the language identification method.

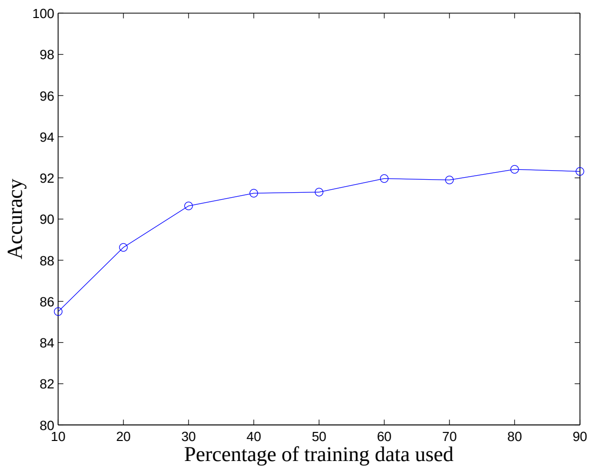


Figure 1: Effect of different amounts of training data on accuracy

In Figure 1 we show the effect of the amount of training data on the accuracy using Logit Boost. We selected Logit Boost for this and the following experiments since its accuracy is comparable to SVMs but it is computationally less expensive. We randomly partitioned the transcription into 10 subgroups. Then we used one subgroup as the test set and the rest for training. Starting with one subgroup in the training set, we incrementally added one subgroup to the training set and evaluate the tagging performance of the test set. We repeated this process several times, choosing randomly a new test set each time. The percentages shown are the average over all the experiments. With only 10% of the sentences for training we are reaching very good accuracy already, as high as that from the strategy based on language identification. The curve flattens after

Table 4: Accuracy of Logit Boost with different subsets of attributes. 'X' marks attributes included. E_t, E_l, E_p, and S_t, S_l, and S_p are the POS tag, lemma and confidence output by the English and the Spanish POS tagger, respectively.

word	E_t	E_l	E_p	S_t	S_l	S_p	**Accuracy**
X	X	X	X	–	–	–	88.80
–	X	X	X	–	–	–	86.22
X	–	–	–	X	X	X	78.59
–	–	–	–	X	X	X	65.28
X	X	X	–	X	X	–	**92.95**
X	X	–	X	X	–	X	92.53
X	X	–	–	X	–	–	91.22
X	X	X	–	–	X	–	89.76
X	–	X	X	–	X	X	77.08
–	–	X	–	–	X	–	74.18
X	X	–	–	–	–	–	85.76
X	–	–	–	–	–	–	71.17
–	–	–	X	–	–	X	24.96
X	X	X	X	X	–	–	92.55
X	X	X	X	–	–	X	88.89
X	–	X	–	X	–	–	78.74
X	X	X	X	–	X	–	89.62
–	X	–	–	X	X	–	90.76
–	–	X	X	–	X	X	75.94
X	–	X	–	X	X	X	80.24
X	–	–	X	X	X	X	79.13

60% of the training data is used. We do not gain much by adding more training data after this.

Results shown in Table 3 demonstrate that POS tagging can be learned effectively based on the attributes described in Subsection 6.1, even if we are not explicitly adding contextual information. To determine the extent to which each attribute is contributing to the learning task, we performed another set of experiments where we selected different subsets of the attributes. Table 4 shows the results with Logit Boost. Overall, the attributes taken from the English POS tagger are more valuable for this learning task. If we only take the word form and the features from the English Tree tagger (first row in Table 4) we are reaching an accuracy that outperforms all heuristics. Still, there is some valuable information provided by the Spanish POS tagger output since the highest accuracy is achieved by including the Spanish-based attributes in combination with the English-based ones. Surprisingly, we can manage to outperform the oracle by using only three attributes: the lexical word form and the POS tags from the English and Spanish tagger (see row 7 in table), or the POS tags from the monolingual taggers together with the lemma from the Spanish tagger (see row 4 from bottom to top). We also experimented adding as an attribute the output of the language identification method, but found no significant changes in the accuracy.

7 Discussion

We analyzed the different results gathered through the experiments and we present here the most relevant insights.

The first discovery, is that a lot of the errors made by the oracle, and the other methods as well, are due to the difficulties inherent in dealing with spontaneous speech where fillers, interruption points, hesitations, and the like abound. About as much as 20% of the errors made by the oracle are due to these features. Another roughly 20% is due to unknown tokens in the transcription, such as mumbling, slang words such as "gonna" and "wanna", or other sounds unintelligible for the human transcriber. For the rest of the analysis we decided to ignore these types of mistakes for all methods and focus only on the remaining mistakes. In the case of the oracle we are left with 445 erroneously POS tagged words. From those, about 50%, or 233 to be exact, are errors in sentences with code-switches. We consider this to be a strong indication of the complexity that intrasentential switches add to the task of POS tagging. For the taggers, these sentences are incomplete, or ill-formed, since they have fragments with foreign words and thus, they fail to identify them. The rest of the oracle mistakes can not be attributable to a single cause. Some are fragmented sentences, and some are due to errors inherent of the tagger, but nothing is particulary salient about them.

The language identification methods share, of course, the same mistakes made by the oracle, plus 342 more, for a total of 787 (in the case of the dictionary-based language identification). The challenge of POS tagging code-switched text is more evident for this method. Out of the mistakes made by the language identification method, 540 lie in sentences with code-switching, that is, nearly 70% of the mistakes. For 307 of these mistakes the right

POS tag was available from one of the taggers. Some typical examples of these errors are words that belong to both languages, such as "a", "no", "me" and "con".

The ML approach outperformed both the language identification method and the oracle. Analyzing the predictions made by SVM we verified that out of the 445 errors made by the oracle, SVM correctly tagged 223, the majority of which are words in sentences with code-switching (142 words). When compared against the errors from the method based on language identification, SVM correctly tagged 481 words out of the 787, 374 of which are words in sentences with code-switches. In summary, the ML approach is more robust to code-switched sentences. Note that we did find some errors made by the ML approach that are not shared by the oracle or the language identification method, a total of 105. Some of these mistakes are due to inconsistencies on the human-annotated tags. For instance, in most cases slang words such as "gonna" and "wanna" are labeled as unknown words, but we found that these words were labeled as verbs in a few cases. Not surprisingly this caused the ML algorithm to fail, since these class labels were misleading. The majority of the mistakes, however, seem to be due to systematic mistakes by the POS taggers.

One last remark is regarding our decision to find a method for successfully exploiting the existing taggers for POS tagging Spanglish text. Our original motivation came from the lack of linguistic resources to process Spanglish text. However, we did train from scratch a sequential model for POS tagging Spanglish, namely Conditional Random Fields (CRFs) (Lafferty *et al.*, 2001). We used MALLET (McCallum, 2002) for this experiment and the same training/testing partitions used in the experiment reported in Table 3. The CRF POS tagger was trained using capitalization information and the previous token as context. The average accuracy of this CRF was 81%, which is lower than the language identification heuristic. We believe that this low accuracy is due to the lack of a representative sample of annotated Spanglish. It will be interesting to see if when more data becomes available the ML algorithms still yield the best results.

8 Conclusions

Code-switching is a fresh and exciting research area that has received little attention in the language processing community. Research on this topic has many interesting applications, including automatic speech recognition, machine translation, and computer assisted language learning. In this paper we present preliminary work towards developing a POS tagger for English-Spanish code-switched text that, to the best of our knowledge, is the first effort towards this end.

We explored different heuristics for taking advantage of existing linguistic resources for English and Spanish with unimpressive results. A simple word-level language identification strategy outperformed all heuristics tested. But the best results, even better than the oracle, were achieved by using machine learning using the output of monolingual POS taggers as input features.

In the error analysis we showed that most of the mistakes made by the language identification method, and the oracle itself, occur in sentences with intrasentential code-switching, showing the difficulty of the task. In contrast, our machine learning approach was less sensitive to the complexity of this alternation pattern.

There is still a lot of work to do in this area. Our ongoing efforts include gathering a larger corpus, with different speakers and conversational styles, as well as written forms of code-switching from blogs and Internet forums. In addition, we are exploring the use of context information. The features we are currently using to represent each word do not take into account the context surrounding the word. We want to test if by using contextual features we can further improve our results.

In this study we focused on code-switching, but borrowing is another complex language alternation pattern that we want the POS tagger to handle. We are working on developing a special method for identification and morphological analysis of borrowings. This method will help increase the accuracy of the POS tagger.

Spanish-English is not the only popular combination of languages. An interesting line of future work would be to explore if the method presented here can be adapted to different language combinations. Moreover, multilingual communities will code-switch among more than two codes and this poses fascinating research challenges as well.

Acknowledgements

We are grateful to Ray Mooney, Melissa Sherman and the four anonymous reviewers for insightful comments and suggestions. Special thanks to Brenda Medina and Nicolle Whitman for helping with some experiments. This research is supported by the National Science Foundation under grant 0812134.

References

Ardila, A. (2005) Spanglish: an anglicized dialect. *Hispanic Journal of Behavioral Sciences*, **27(1)**: 60–81.

Brants, T. (2000) TnT - a statistical part-of-speech tagger. In *Sixth Applied Natural Language Processing Conference ANLP-2000* Seatle, WA.

Brill, E. (1990) A simple rule-based part-of-speech tagger. In *3rd Conference on Applied Natural Language Processing*, pp. 152–155. Trento, Italy.

Carreras, X. and Padró, L. (2002) A flexible distributed architecture for natural language analyzers. In *Third International Conference on Language Resources and Evaluation, LREC-02*, pp. 1813–1817. Las Palmas de Gran Canaria, Spain.

Charniak, E. (1993) Equations for part-of-speech tagging. In *11th National Conference on AI*, pp. 784–789.

Ervin, S. and Osgood, C. (1954) Second language learning and bilingualism. *Journal of abnormal and social phsychology*, Supplement 49, pp. 139–146.

Espinosa, A. (1917) Speech mixture in New Mexico: the influence of English language on New Mexican Spanish. In H. Stevens and H. Bolton, (eds.), *The Pacific Ocean in History*, pp. 408–428.

Franco, J.C. and Solorio T. (2007) Baby-steps towards building a Spanglish language model. In *8th International Conference on Computational Linguistics and Intelligent Text Processing, CICLing-2007*, pp. 75–84. Mexico City, Mexico.

Friedman, J., Hastie, T., and Tibshirani, R. (1998) Additive logistic regression: a statistical view of boosting. Technical Report, Stanford University.

Godfrey, J., Holliman, E. and McDaniel, J. (1992) Switchboard: Telephone speech corpus for research development. In *ICASSP*, pp. 517–520, San Francisco, CA, USA.

Goyal, P., Mital, Manav R., Mukerjee, A., Raina, Achla M., Sharma, D., Shukla, P. and Vikram, K. (2003) A bilingual parser for Hindi, English and code-switching structures. In *Computational Linguistics for South Asian Languages –Expanding Synergies with Europe, EACL-2003 Workshop*, Budapest, Hungary.

Grosjean, F. (1982) *Life with Two Languages: An Introduction to Bilingualism*. Harvard University Press.

Gumperz, J. J. and Hernandez-Chavez, E. (1971) *Cognitive aspects of bilingual communication*. Oxford University Press, London.

Gumperz, J. J. (1964) Linguistic and social interaction in two communities. In John J. Gumperz (ed.), *Language in social groups*, pp. 151–176, Stanford. Stanford University Press.

Gumperz, J. J. (1971) Bilingualism, bidialectism and classroom interaction. In *Language in social groups*, pp. 311–339, Stanford. Stanford University Press.

Joshi, A. K. (1982) Processing of sentences with intrasentential code-switching. In Ján Horecký (ed.), *COLING-82*, pp. 145–150, Prague.

Kucera, H. and Francis, W. N. (1967) *Computational analysis of present-day American English*. Brown University Press.

Lafferty, J. and McCallum, A. and Pereira F. (2001) Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *18th ICML*, pp. 282–289. MA, USA.

Lipski, J. M. (1978) Code-switching and the problem of bilingual competence. In M. Paradis (ed.), *Aspects of bilingualism*, pp. 250–264, Columbia, SC. Hornbeam.

McCallum, A. (2002) MALLET: A Machine Learning for Language Toolkit. Retrieved January 7, 2008 from `http://mallet.cs.umass.edu`

Paul, D. B. and Baker, J. M. (1992) The design of the Wall Street Journal-based CSR corpus. In *HLT'91: workshop on speech and Natural Language* pp. 357–362, Morristown, NJ, USA.

Poplack, S., Sankoff, D. and Miller, C. (1988) The social correlates and linguistic processes of lexical borrowing and assimilation. *Linguistics*, **26(1)**: 47–104.

Poplack, S. (1980) Sometimes I'll start a sentence in Spanish y termino en español: toward a typology of code-switching. *Linguistics*, **18(7/8)**: 581–618.

Poplack, S. (1981) Syntactic structure and social function of code-switching. In R. Duran (ed.), *Latino discourse and communicative behavior*, pp. 169–184, Norwood, NJ. Ablex.

Quinlan, J. R. (1986) Induction of decision trees. *Machine Learning*, **1**: 81–106.

Ratnaparkhi, A. (1996) A maximum entropy model for part-of-speech tagging. In *EMNLP*, pp. 133–142, Philadelphia, PA, May.

Rosner, M. and Farrugia, P. (2007) A tagging algorithm for mixed language identification in a noisy domain.

In *INTERSPEECH 2007*, pp. 190–193, Antwerp, Belgium.

Sankoff, D. (1968) Social aspects of multilingualism in New Guinea. Ph.D. thesis, McGill University.

Sankoff, D. (1981) A formal grammar for codeswitching. *Papers in Linguistics: International Journal of Human communications*, **14(1)**: 3–46.

Sankoff, D. (1998a) A formal production-based explanation of the facts of code-switching. *Bilingualism, Language and Cognition*, **1**: 39–50. Cambridge University Press.

Sankoff, D. (1998b) The production of code-mixed discourse. In *36th ACL*, volume I, pp. 8–21, Montreal, Quebec, Canada.

Schmid, H. (1994) Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing, Manchester, UK*.

Schölkopf, B. and Smola, A. J. (2002) *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press.

Toribio, A. J. (2001a) Accessing Spanish-English code-switching competence. *International Journal of Bilingualism*, **5(4)**:403–436.

Toribio, A. J. (2001b) On the emergence of bilingual code-switching competence. *Bilingual Language and Cognition*, **4(3)**:203–231.

U.S. Census Bureau. (2003) Language use and English speaking ability: 2000. Retrieved October 30, 2006 from `http://www.census.gov/prod/2003pubs/c2kbr-29.pdf`.

Witten, I. H. and Frank, E. (1999) *Data Mining, Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann.

Wolpert, D. H. (1992) Stacked Generalization. *Neural Networks*, **5(2)**:241–259.

Information Retrieval Oriented Word Segmentation based on Character Associative Strength Ranking

Yixuan Liu, Bin Wang, Fan Ding, Sheng Xu
Information Retrieval Group
Center for Advanced Computing Research
Institute of Computing Technology
Chinese Academy of Sciences
Beijing, 100190, P.R.China
{liuyixuan, wangbin, dingfan, xusheng}@ict.ac.cn

Abstract

This paper presents a novel, ranking-style word segmentation approach, called *RSVM-Seg*, which is well tailored to Chinese information retrieval(CIR). This strategy makes segmentation decision based on the ranking of the internal associative strength between each pair of adjacent characters of the sentence. On the training corpus composed of query items, a ranking model is learned by a widely-used tool Ranking SVM, with some useful statistical features, such as mutual information, difference of t-test, frequency and dictionary information. Experimental results show that, this method is able to eliminate overlapping ambiguity much more effectively, compared to the current word segmentation methods. Furthermore, as this strategy naturally generates segmentation results with different granularity, the performance of CIR systems is improved and achieves the state of the art.

1 Introduction

To improve information retrieval systems' performance, it is important to comprehend both queries and corpus precisely. Unlike English and other western languages, Chinese does not delimit words by white-space. Word segmentation is therefore a key preprocessor for Chinese information retrieval to comprehend sentences.

Due to the characteristics of Chinese, two main problems remain unresolved in word segmentation: segmentation ambiguity and unknown words, which are also demonstrated to affect the performance of Chinese information retrieval (Foo and Li, 2004). Overlapping ambiguity and combinatory ambiguity are two forms of segmentation ambiguity. The first one refers to that ABC can be segmented into AB C or A BC. The second one refers to that string AB can be a word, or A can be a word and B can be a word. In CIR, the combinatory ambiguity is also called segmentation granularity problem (Fan et al., 2007). There are many researches on the relationship between word segmentation and Chinese information retrieval (Foo and Li, 2004; Peng et al., 2002a; Peng et al., 2002b; Jin and Wong, 2002). Their studies show that the segmentation accuracy does not monotonically influence subsequent retrieval performance. Especially the overlapping ambiguity, as shown in experiments of (Wang, 2006), will cause more performance decrement of CIR. Thus a CIR system with a word segmenter better solving the overlapping ambiguity, may achieve better performance. Besides, it also showed that the precision of new word identification was more important than the recall.

There are some researches show that when compound words are split into smaller constituents, better retrieval results can be achieved (Peng et al., 2002a). On the other hand, it is reasonable that the longer the word which co-exists in query and corpus, the more similarity they may have. A hypothesis, therefore, comes to our mind, that different segmentation granularity can be incorporated to obtain better CIR performance.

In this paper we present a novel word segmentation approach for CIR, which can not only obviously reduce the overlapping ambiguity, but also introduce different segmentation granularity for the first time.

In our method, we first predict the ranking result of all internal association strength (IAS) between each pair of adjacent characters in a sentence using Ranking SVM model, and then, we segment the sentence into sub-sentences with smaller and smaller granularity by cutting adjacent character pairs according to this rank. Other machine-learning based segmentation algorithms (Zhang et al., 2003; Lafferty et al., 2001; Ng and Low, 2004) treat segmentation problem as a character sequence tagging problem based on classification. However, these methods cannot directly obtain different segmentation granularity. Experiments show that our method can actually improve information retrieval performance.

This paper is structured as follows. It starts with a brief introduction of the related work on the word segmentation approaches. Then in Section 3, we introduce our segmentation method. Section 4 evaluates the method based on experimental results. Finally, Section 5 makes summary of this whole paper and proposes the future research orientation.

2 Related Work

Various methods have been proposed to address the word segmentation problem in previous studies. They fall into two main categories, rule-based approaches that make use of linguistic knowledge and statistical approaches that train on corpus with machine learning methods. In rule-based approaches, algorithms of string matching based on dictionary are the most commonly used, such as maximum matching. They firstly segment sentences according to a dictionary and then resort to some rules to resolve ambiguities (Liu, 2002; Luo and Song, 2001). These rule-based methods are fast, however, their performances depend on the dictionary which cannot include all words, and also on the rules which cost a lot of time to make and must be updated frequently. Recent years statistical approaches became more popular. These methods take advantage of various probability information gained from large corpus to segment sentences. Among them, Wang's work (Wang, 2006) is the most similar to our method, since both of us apply statistics information of each gap in the sentence to eliminate overlapping ambiguity in methods. However, when combining different statistics, Wang decided the weight by a heuristic way which was too simply to be suitable for all sentences. In our method, we employ a machine-learning method to train features' weights.

Many machine-learning methods, such as HMM (Zhang et al., 2003), CRF (Lafferty et al., 2001), Maximum Entropy (Ng and Low, 2004), have been exploited in segmentation task. To our knowledge, machine-learning methods used in segmentation treated word segmentation as a character tagging problem. According to the model trained from training corpus and features extracted from the context in the sentence, these methods assign each character a positional tag, indicating its relative position in the word. These methods are difficult to get different granularity segmentation results directly. Our method has two main differences with them. Firstly, we tag the gap between characters rather than characters themselves. Secondly, our method is based on ranking rather than classification.

Then, we will present our ranking-based segmentation method, *RSVM-Seg*.

3 Ranking based Segmentation

Traditional segmentation methods always take the segmentation problem as classification problem and give a definite segmentation result. In our approach, we try to solve word segmentation problem from the view of ranking. For easy understanding, let's represent a Chinese sentence S as a character sequence:

$$C_{1:n} = C_1C_2\dots C_n$$

We also explicitly show the gap $G_i(i = 1\dots n-1)$ between every two adjacent characters C_i and C_{i+1}:

$$C_{1:n}|G_{1:n-1} = C_1G_1C_2G_2\dots G_{n-1}C_n$$

$IAS_i(i = 1\dots n)$ is corresponding to $G_i(i = 1\dots n)$, reflecting the internal association strength between C_i and C_{i+1}. The higher the IAS value is, the stronger the associative between the two characters is. If the association between two characters is weak, then they can be segmented. Otherwise, they should be unsegmented. That is to say we could make segmentation based on the ranking of IAS value. In our ranking-style segmentation method, Ranking SVM is exploited to predict IAS ranking.

In next subsections, we will introduce how to take advantage of Ranking SVM model to solve our problem. Then, we will describe features used for training the Ranking SVM model. Finally, we will give a scheme how to get segmentation result from predicted ranking result of Ranking SVM.

3.1 Segmentation based on Ranking SVM

Ranking SVM is a classical algorithm for ranking, which formalizes learning to rank as learning for classification on pairs of instances and tackles the classification issue by using SVM (Joachims, 2002). Suppose that $X \epsilon R^d$ is the feature space, where d is the number of features, and $Y = r_1, r_2, \ldots, r_K$ is the set of labels representing ranks. And there exists a total order between ranks $r_1 > r_2 > \ldots > r_K$, where $>$ denotes the order relationship. The actual task of learning is formalized as a Quadratic Programming problem as shown below:

$$\begin{aligned} & min_{\omega,\xi_{ij}} \frac{1}{2}\|\omega\|^2 + C\Sigma\xi_{ij} \\ & s.t.\langle \omega, x_i - x_j \rangle > 1 - \xi_{ij}, \forall x_i \succ x_j, \xi_{ij} \geq 0 \end{aligned} \quad (1)$$

where $\|\omega\|$ denotes l_2 norm measuring the margin of the hyperplane and ξ_{ij} denotes a slack variable. $x_i \succ x_j$ means the rank class of x_i has an order prior to that of x_j, i.e. $Y(x_i) > Y(x_j)$. Suppose that the solution to (1) is ω_*, then we can make the ranking function as $f(x) = \langle \omega_*, x \rangle$.

When applying Ranking SVM model to our problems, an instance (feature vector x) is created from all bigrams (namely $C_iC_{i+1}, i = 1 \ldots n-1$) of a sentence in the training corpus. Each feature is defined as a function of bigrams (we will describe features in detail in next subsection). The instances from all sentences are then combined for training. And Y refers to the class label of the IAS degree. As we mentioned above, segmentation decision is based on IAS value. Therefore, the number of IAS degree's class label is also correspondent to the number of segmentation class label. In traditional segmentation algorithms, they always label segmentation as two classes, segmented and unsegmented. However, for some phrases, it is a dilemma to make a segmentation decision based on this two-class scheme. For example, Chinese phrase "笔记本电脑(Notepad)" can be segmented as "笔记本(Note)" and "电脑(computer)" or can be viewed as one word. We cannot easily classify the gap between "本" and "脑" as segmented or unsegmented. Therefore, beside these two class labels, we define another class label, semisegmented, which means that the gap between two characters could be segmented or unsegmented, either will be right. Correspondingly, IAS degree is also divided into three classes, definitely inseparable (marked as 3), partially inseparable (marked as 2), and separable (marked as 1). "Separable" corresponds to be segmented"; "partially inseparable" corresponds to semisegmented; "definitely inseparable" corresponds to be unsegmented. Obviously, there exists orders between these labels' IAS values, namely $IAS(1) < IAS(2) < IAS(3), IAS(*)$ represents the IAS value of different labels. Next, we will describe the features used to train Ranking SVM model.

3.2 Features for IAS computation

Mutual Information: Mutual information, measuring the relationship between two variables, has been extensively used in computational language research. Given a Chinese character string 'xy' (as mentioned above, in our method, 'xy' refers to bigram in a sentence), mutual information between characters x and y is defined as follows:

$$mi(x,y) = log_2 \frac{p(x,y)}{p(x)p(y)} \quad (2)$$

where $p(x,y)$ is the co-occurrence probability of x and y, namely the probability that bigram 'xy' occurs in the training corpus, and $p(x), p(y)$ are the independent probabilities of x and y respectively. From (2), we conclude that $mi(x,y) \gg 0$ means that IAS is strong; $mi(x,y) \approx 0$ means that it is indefinite for IAS between characters x and y; $mi(x,y) \ll 0$ means that there is no association been characters x and y. However, mutual information has no consideration of context, so it cannot solve the overlapping ambiguity effectively (Sili Wang 2006). To remedy this defect, we introduce another statistics measure, difference of t-test.

Difference of t-score (DTS)**:** Difference of t-score is proposed on the basis of t-score. Given a Chinese character string 'xyz', the t-score of the character y relevant to character x and z is defined as:

$$t_{x,z}(y) = \frac{p(z|y) - p(y|x)}{\sqrt{\sigma_2(p(z|y)) + \sigma_2(p(y|x))}} \quad (3)$$

where $p(y|x)$ is the conditional probability of y given x, and $p(z|y)$, of z given y, and $\sigma_2(p(y|x))$, $\sigma_2(p(z|y))$ are variances of $p(y|x)$ and of $p(z|y)$ respectively. Sun et al. gave the derivation formula of $\sigma_2(p(y|x)), \sigma_2(p(z|y))$ (Sun et al., 1997) as

$$\sigma_2(p(z|y)) \approx \frac{r(y,z)}{r^2(y)} \quad \sigma_2(p(y|x)) \approx \frac{r(x,y)}{r^2(x)} \quad (4)$$

where $r(x,y)$, $r(y,z)$, $r(y)$, $r(z)$ are the frequency of string xy, yz, y, and z respectively. Thus formula (3) is deducted as

$$t_{x,z}(y) = \frac{\frac{r(y,z)}{r(y)} - \frac{r(x,y)}{r(x)}}{\sqrt{\frac{r(y,z)}{r^2(y)} + \frac{r(x,y)}{r^2(x)}}} \quad (5)$$

$t_{x,z}(y)$ indicates the binding tendency of y in the context of x and z: if $t_{x,z}(y) > 0$ then y tends to be bound with z rather than with x; if $t_{x,z}(y) < 0$, they y tends to be bound with x rather than with z.

To measure the binding tendency between two adjacent characters 'xy' (also, it refers to bigram in a sentence in our method), we use difference of t-score (DTS) (Sun et al., 1998) which is defined as

$$dts(x,y) = t_{v,y}(x) - t_{x,w}(y) \quad (6)$$

Higher $dts(x,y)$ indicates stronger IAS between adjacent characters x and y.

Dictionary Information: Both statistics measures mentioned above cannot avoid sparse data problem. Then Dictionary Information is used to compensate for the shortage of statistics information. The dictionary we used includes 75784 terms. We use binary value to denote the dictionary feature. If a bigram is in the dictionary or a part of dictionary term, we label it as "1", otherwise, we label is as "0".

Frequency: An important characteristic of new word is its repeatability. Thus, we also use frequency as another feature to train Ranking SVM model. Here, the frequency is referred to the number of times that a bigram occurs in the training corpus.

We give a training sentence for a better understanding of features mentioned above. The sentence is "中国建设银行网(China Construction Bank network)" We extract all bigrams in this sentence, compute the four above features and give the IAS a label for each bigram. The feature vectors of all these bigrams for training are shown in Table 1.

Algorithm 1 : Generate various granularity terms

```
1:  Input: A Chinese sentence $S = C_1 : C_n$
    $IAS = IAS_{1:n-1}$ $LB = 1; RB = n$
2:  $Iterative(S, IAS)$:
3:  while $length(S) \geq 3$ do
4:      $MB = FindMinIAS(IAS)$
5:      $SL = C_{LB:MB}$
6:      $SR = C_{MB+1:RB}$
7:      $IAS_L = IAS_{LB:MB}$
8:      $IAS_R = IAS_{MB+1:RB}$
9:      $Iterative(SL, IAS_L)$
10:     $Iterative(SR, IAS_R)$
11: end while
```

3.3 Segmentation scheme

In order to compare with other segmentation methods, which give a segmentation result based on two class labels, segmented and unsegmented, it is necessary to convert real numbers result given by Ranking SVM to these two labels. Here, we make a heuristic scheme to segment the sentence based on IAS ranking result predicted by Ranking SVM. The scheme is described in Algorithm 1. In each iteration we cut the sentence at the gap with minimum IAS value. Nie et.al. pointed out that the average length of words in usage is 1.59 (Nie et al., 2000). Therefore, we stop the segmentation iterative when the length of sub_sentence is 2 or less than 2. By this method, we could represent the segmentation result as a binary tree. Figure 1 shows an example of this tree. With this tree, we can obtain various granularity segmentations easily, which could be used in CIR. This segmentation scheme may cause some combinatory ambiguity. However, Nie et.al. (Nie et al., 2000) also pointed out that there is no accurate word definition, thus whether combinatory ambiguity occurs is uncertain. What's more, compared to overlapping ambiguity, combinatory ambiguity is not the fatal factor for information retrieval performance as mentioned in introduction. Therefore, this scheme is reasonable for Chinese information re-

Bigram	MI	DTS	Dictionary	Frequency	IAS
中国(China)	6.67	1985.26	1	1064561	3
国建	2.59	-1447.6	0	14325	1
建设(Construction)	8.67	822.64	1	200129	3
设银	5.94	-844.05	0	16098	2
银行(Bank)	9.22	931.25	1	236976	3
行网	2.29	-471.24	0	15282	1

Table 1: Example of feature vector

江西省交通地图
(Traffic map of JiangXi Province)

江西省		交通地图	
(JiangXi Province)		(Traffic map)	
江西	省	交通	地图
(JiangXi)	(Province)	(Traffic)	(Map)

Figure 1: Example 1

trieval.

4 Experiments and analysis

4.1 Data

Since the label scheme and evaluation measure (described in next subsection) of our segmentation method are both different from the traditional segmentation methods, we did not carry out experiments on SIGHAN. Instead, we used two query logs (QueryLog1 and QueryLog2) as our experiment corpus, which are from two Chinese search engine companies. 900 queries randomly from QueryLog1 were chosen as training corpus. 110 Chinese queries from PKU Tianwang[1] , randomly selected 150 queries from QueryLog1 and 100 queries from QueryLog2 were used as test corpus. The train and test corpus have been tagged by three people. They were given written information need statements, and were asked to judge the IAS of every two adjacent characters in a sentence on a three level scale as mentioned above, separable, partially inseparable, and definitely inseparable. The assessors agreed in 84% of the sentences, the other sentences were checked by all assessors, and a more plausible alternative was selected. We exploited SVM^{light2} as the toolkit to implement Ranking SVM model.

4.2 Evaluation Measure

Since our approach is based on the ranking of IAS values, it is inappropriate to evaluate our method by the traditional method used in other segmentation algorithms. Here, we proposed an evaluation measure RankPrecision based on Kendall's τ (Joachims, 2002), which compared the similarity between the predicted ranking of IAS values and the rankings of these tags as descending order. RankPrecision formula is as follows:

$$RankPrecision = 1 - \frac{\Sigma_{i=1}^{n} InverseCount(s_i)}{\Sigma_{i=1}^{n} CompInverseCount(s_i)} \quad (7)$$

where s_i represents the ith sentence (unsegmented string), $InverseCount(s_i)$ represents the number of discordant pairs inversions in the ranking of the predicted IAS value compared to the correct labeled ranking. $CompInverseCount(s_i)$ represents the number of discordant pairs inversions when the labels totally inverse.

4.3 Experiments Results

Contributions of the Features: We investigated the contribution of each feature by generating many versions of Ranking SVM model. RankPrecision as described above was used for evaluations in these and following experiments. We used Mutual Information(MI); Difference of T-Score(DTS); Frequency(F); mutual information and difference of t-score(MI+DTS); mu-

[1]Title field of SEWM2006 and SEWM2007 web retrieval TD task topics. See http://www.cwirf.org/

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

Feature	Corpus			
	Train	Query Log1	Query Log2	Tian Wang
MI	0.882	0.8719	0.8891	0.9444
DTS	0.9054	0.8954	0.9086	0.9444
F	0.8499	0.8416	0.8563	0.9583
MI+DTS	0.9077	0.9117	0.923	0.9769
MI+DTS+F	0.8896	0.8857	0.9209	0.9815
MI+DTS+D	**0.933**	0.916	**0.9384**	**0.9954**
MI+DTS+F+D	0.932	**0.93**	0.9374	**0.9954**

Table 2: The segmentation performance with different features

Size of Train Corpus	Corpus			
	Train	Query Log1	Query Log2	Tian Wang
100	0.9149	0.9070	0.9209	0.9630
200	0.9325	0.9304	0.9446	0.9907
400	0.9169	0.9057	0.9230	0.9630
500	0.9320	0.9300	0.9374	0.9954
600	0.9106	0.9050	0.9312	0.9907
700	0.9330	0.9284	0.9353	0.9954
900	0.9217	0.9104	0.9240	0.9907

Table 3: The segmentation performance with different size training corpus

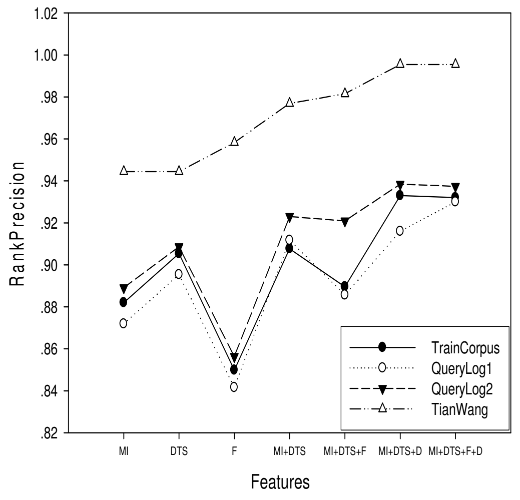


Figure 2: Effects of features

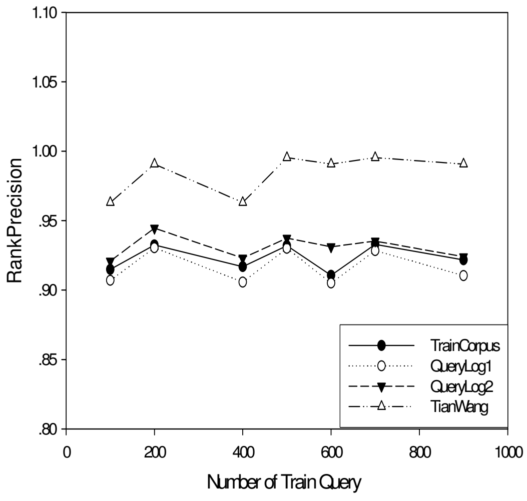


Figure 3: Effects of Corpus Size

tual information, difference of t-score and Frequency(MI+DTS+F); mutual information, difference of t-score and dictionary(MI+DTS+D); mutual information, difference of t-score, frequency and Dictionary(MI+DTS+F+D) as features respectively. The results are shown in Table 2 and Figure 2. From the results, we can see that:

- Using all described features together, the Ranking SVM achieved a good performance. And when we added MI, DTS, frequency, dictionary as features one by one, the RankPrecision improved step by step. It demonstrates that the features we selected are useful for segmentation.
- The lowest RankPrecision is above 85%, which suggests that the predicted rank result by our approach is very close to the right rank. It is shown that our method is effective.
- When we used each feature alone, difference of t-score achieved highest RankPrecise, frequency was worst on most of test corpus (except TianWang). It is induced that difference of t-test is the most effective feature for segmentation. It is explained that because dts is combined with the context information, which eliminates overlapping ambiguity errors.
- It is surprising that when mutual information and difference of t-score was combined with

frequency, the RankPrecision was hurt on three test corpus, even worse than dts feature. The reason is supposed that some non-meaning but common strings, such as ”的人” would be took for a word with high IAS values. To correct this error, we could build a stop word list, and when we meet a character in this list, we treat them as a white-space.

Effects of corpus size: We trained different Ranking SVM models with different corpus size to investigate the effects of training corpus size to our method performance. The results are shown in Table 3 and Figure 3. From the results, we can see that the effect of corpus size to the performance of our approach is minors. Our segmentation approach can achieve good performance even with small training corpus, which indicates that Ranking SVM has generalization ability. Therefore we can use a relative small corpus to train Ranking SVM, saving labeling effort.

Effects on Finding Boundary: In algorithm 1, we could get different granularity segmentation words when we chose different length as stop condition. Figure 4 shows the ”boundary precision” at each stop condition. Here, ”boundary precision” is defined as

$$\frac{No.of\ right\ cut\ boundaries}{No.of\ all\ cut\ boundaries} \tag{8}$$

From the result shown in figure 4, we can see that as the segmentation granularity gets smaller, the boundary precision gets lower. The reason is obvious, that we may segment a whole word into smaller parts. However, as we analyzed in introduction, in CIR, we should judge words boundaries correctly to avoid overlapping ambiguity. As for combinatory ambiguity, through setting different stop length condition, we can obtain different granularity segmentation result.

Effects on Overlapping Ambiguity: Due to the inconsistency of train and test corpus, it is difficult to keep fair for Chinese word segmentation evaluation. Since ICTCLAS is considered as the best Chinese word segmentation systems. We chose ICTCLAS as the comparison object. Moreover, we chose Maximum Match segmentation algorithm, which is rule-based segmentation method, as the baseline.

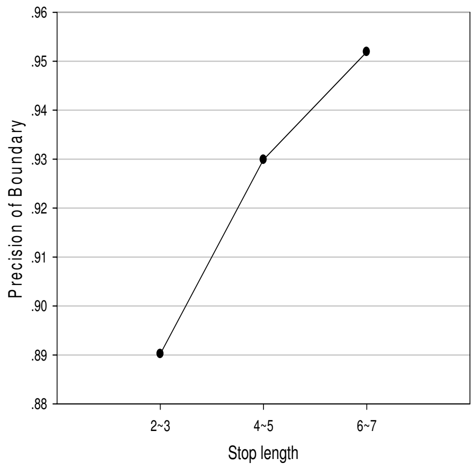


Figure 4: Precision of boundary with different stop word length conditions

Corpus	**NOA** (RSVM_Seg)	**NOA** (ICTCLAS)	**NOA** (MM)
Query Log1	7	10	21
Query Log2	2	6	16
Tian Wang	0	0	1

Table 4: Number of Overlapping Ambiguity

We compared the number of overlapping ambiguity(NOA) among these three approaches on test corpus QueryLog1, QueryLog2 and TianWang. The result is shown in Table 4. On these three test corpus, the NOA of our approach is smallest, which indicates our method resolve overlapping ambiguity more effectively. For example, the sentence ”基础课件(basic notes)”, the segmentation result of ICTCLAS is ”基础课(basic class)/件(article)”, the word ”课件(notes)” is segmented, overlapping ambiguity occurring. However, with our method, the predicted IAS value rank of positions between every two adjacent characters in this sentence is ”基3础1课2件”, which indicates that the character ”课” has stronger internal associative strength with the character ”件” than with the character ”础”, eliminating overlapping ambiguity according to this ISA rank results.

Effects on Recognition Boundaries of new word: According to the rank result of all IAS values

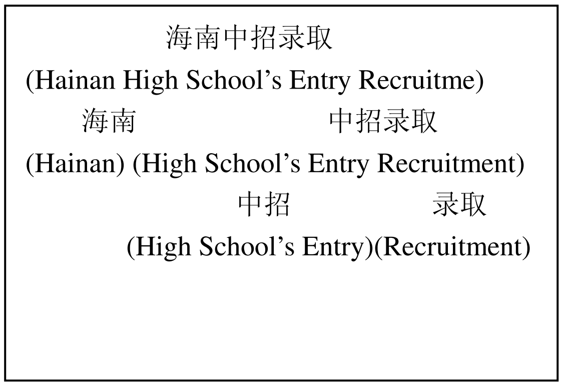

海南中招录取
(Hainan High School's Entry Recruitme)
海南
中招录取
(Hainan) (High School's Entry Recruitment)
中招
录取
(High School's Entry)(Recruitment)

Figure 5: Example of New Word boundary

in a sentence, our method can recognize the boundaries of new words precisely, avoiding the overlapping ambiguity caused by new words. For example, the phrase "海南中招录取(Hainan High School's Entry Recruitment)", the ICTCLAS segmentation result is "海南/中/招录/取", because the new word "中招" cannot be recognized accurately, thus the character "招" is combined with its latter character "录", causing overlapping ambiguity. By our method, the segmentation result is shown as figure 5, in which no overlapping ambiguity occurs.

Performance of Chinese Information Retrieval: To evaluate the effectiveness of *RSVM-Seg* method on CIR, we compared it with the FMM segmentation. Our retrieval system combines different query representations obtained by our segmentation method, *RSVM-Seg*. In previous TREC Terebyte Track, Markov Random Field(MRF) (Metzler and Croft, 2005) model has displayed better performance than other information retrieval models, and it can much more easily include dependence features. There are three variants of MRF model, full independence(FI), sequential dependence(SD), and full dependence(FD). We chose SD as our retrieval model, since Chinese words are composed by characters and the adjacent characters have strong dependence relationship. We evaluated the CIR performance on the Chinese Web Corpora CWT200g provided by Tianwang [3], which, as we know, is the largest publicly available Chinese web corpus till now. It consists of $37,482,913$ web pages with total size of 197GB. We used the topic set for SEWM2007 and SEWM2006 Topic Distillation (TD) task which contains 121 topics. MAP, R-Precision and GMAP (Robertson, 2006) were as main evaluation metrics. GMAP is the geometric mean of AP(Average Precision) through different queries, which was introduced to concentrate on difficult queries. The result is shown in 5. From the table, we can see that our segmentation method improve the CIR performance compared to FMM.

[3]http://www.cwirf.org/

Segmentation Method	MAP	R-P	GMAP
FMM	0.0548	0.0656	0.0095
RSVM-Seg	0.0623	0.0681	0.0196

Table 5: Evaluation of CIR performance

5 Conclusion and Future work

From what we have discussed above, we can safely draw the conclusion that our work includes several main contributions. Firstly, to our best known, this is the first time to take the Chinese word segmentation problem as ranking problem, which provides a new view for Chinese word segmentation. This approach has been proved to be able to eliminate overlapping ambiguity and also be able to obtain various segmentation granularities. Furthermore, our segmentation method can improve Chinese information retrieval performance to some extent.

As future work, we would search another more encouraging method to make a segmentation decision from the ranking result. Moreover, we will try to relabel SIGHAN corpus on our three labels, and do experiments on them, which will be more convenient to compare with other segmentation methods. Besides, we will carry out more experiments to search the effectiveness of our segmentation method to CIR.

Acknowledgments

This paper is supported by China Natural Science Founding under No. 60603094 and China National 863 key project under No. 2006AA010105. We appreciate Wenbin Jiang's precious modification advices. Finally, we would like to thank the three anonymous EMNLP reviewers for their helpful and constructive comments.

References

D. Fan, W. Bin, and W. Sili. 2007. A Heuristic Approach for Segmentation Granularity Problem in Chinese Information Retrieval. *Advanced Language Processing and Web Information Technology, 2007. ALPIT 2007. Sixth International Conference on*, pages 87–91.

S. Foo and H. Li. 2004. Chinese word segmentation and its effect on information retrieval. volume 40, pages 161–190. Elsevier.

H. Jin and K.F. Wong. 2002. A Chinese dictionary construction algorithm for information retrieval. *ACM Transactions on Asian Language Information Processing (TALIP)*, 1(4):281–296.

T. Joachims. 2002. Optimizing search engines using clickthrough data. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142.

J.D. Lafferty, A. McCallum, and F.C.N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. *Proceedings of the Eighteenth International Conference on Machine Learning table of contents*, pages 282–289.

Q. Liu. 2002. Review of Chinese lexical and syntactic technology.

Z.Y. Luo and R. Song. 2001. Proper noun recognition in Chinese word segmentation research. *Conference of international Chinese computer*, 328:2001–323.

D. Metzler and W.B. Croft. 2005. A Markov random field model for term dependencies. *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 472–479.

H.T. Ng and J.K. Low. 2004. Chinese part-of-speech tagging: one-at-a-time or all-at-once? word-based or character-based. *Proc of EMNLP*.

J.Y. Nie, J. Gao, J. Zhang, and M. Zhou. 2000. On the use of words and n-grams for Chinese information retrieval. *Proceedings of the fifth international workshop on on Information retrieval with Asian languages*, pages 141–148.

F. Peng, X. Huang, D. Schuurmans, and N. Cercone. 2002a. Investigating the relationship between word segmentation performance and retrieval performance in Chinese IR. *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7.

F. Peng, X. Huang, D. Schuurmans, N. Cercone, and S.E. Robertson. 2002b. Using self-supervised word segmentation in Chinese information retrieval. *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 349–350.

S. Robertson. 2006. On GMAP: and other transformations. *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 78–83.

Sili Wang. 2006. Research on chinese word segmentation for large scale information retrieval.

H.P. Zhang, H.K. Yu, D.Y. Xiong, and Q. Liu. 2003. HHMM-based Chinese Lexical Analyzer ICTCLAS. *Proceedings of Second SIGHAN Workshop on Chinese Language Processing*, pages 184–187.

An Analysis of Active Learning Strategies for Sequence Labeling Tasks

Burr Settles[*†]
[*]Dept. of Computer Sciences
University of Wisconsin
Madison, WI 53706, USA
bsettles@cs.wisc.edu

Mark Craven[†*]
[†]Dept. of Biostatistics & Medical Informatics
University of Wisconsin
Madison, WI 53706, USA
craven@biostat.wisc.edu

Abstract

Active learning is well-suited to many problems in natural language processing, where unlabeled data may be abundant but annotation is slow and expensive. This paper aims to shed light on the best active learning approaches for sequence labeling tasks such as information extraction and document segmentation. We survey previously used query selection strategies for sequence models, and propose several novel algorithms to address their shortcomings. We also conduct a large-scale empirical comparison using multiple corpora, which demonstrates that our proposed methods advance the state of the art.

1 Introduction

Traditional supervised learning algorithms use whatever labeled data is provided to induce a model. By contrast, *active learning* gives the learner a degree of control by allowing it to select which instances are labeled and added to the training set. A typical active learner begins with a small labeled set $\mathcal{L}$, selects one or more informative *query* instances from a large unlabeled pool $\mathcal{U}$, learns from these labeled queries (which are then added to $\mathcal{L}$), and repeats. In this way, the learner aims to achieve high accuracy with as little labeling effort as possible. Thus, active learning can be valuable in domains where unlabeled data are readily available, but obtaining training labels is expensive.

Such is the case with many *sequence labeling* tasks in natural language domains. For example, part-of-speech tagging (Seung et al., 1992; Lafferty et al., 2001), information extraction (Scheffer et al., 2001; Sang and DeMeulder, 2003; Kim et al., 2004), and document segmentation (Carvalho and Cohen, 2004) are all typically treated as sequence labeling problems. The source data for these tasks (i.e., text documents in electronic form) are often easily obtained. However, due to the nature of sequence labeling tasks, annotating these texts can be rather tedious and time-consuming, making active learning an attractive technique.

While there has been much work on active learning for classification (Cohn et al., 1994; McCallum and Nigam, 1998; Zhang and Oles, 2000; Zhu et al., 2003), active learning for sequence labeling has received considerably less attention. A few methods have been proposed, based mostly on the conventions of *uncertainty sampling*, where the learner queries the instance about which it has the least certainty (Scheffer et al., 2001; Culotta and McCallum, 2005; Kim et al., 2006), or *query-by-committee*, where a "committee" of models selects the instance about which its members most disagree (Dagan and Engelson, 1995). We provide more detail on these and the new strategies we propose in Section 3.

The comparative effectiveness of these approaches, however, has not been studied. Furthermore, it has been suggested that uncertainty sampling and query-by-committee fail on occasion (Roy and McCallum, 2001; Zhu et al., 2003) by querying outliers, e.g., instances considered informative in isolation by the learner, but containing little information about the *rest* of the distribution of instances. Proposed methods for dealing with these shortcomings have so far only considered classification tasks.

This paper presents two major contributions for active learning and sequence labeling tasks. First, we motivate and introduce several new query strategies for probabilistic sequence models. Second, we conduct a thorough empirical analysis of previously proposed methods with our algorithms on a variety of benchmark corpora. The remainder of this paper is organized as follows. Section 2 provides a brief introduction to sequence labeling and conditional random fields (the sequence model used in our experiments). Section 3 describes in detail all the query selection strategies we consider. Section 4 presents the results of our empirical study. Section 5 concludes with a summary of our findings.

2 Sequence Labeling and CRFs

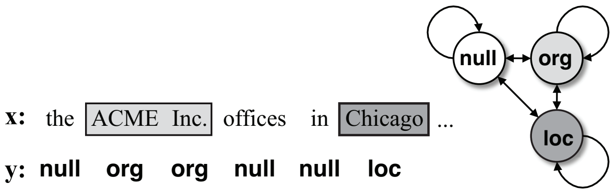
In this paper, we are concerned with active learning for sequence labeling. Figure 1 illustrates how, for example, an information extraction problem can be viewed as a sequence labeling task. Let $\mathbf{x} = \langle x_1, \ldots, x_T \rangle$ be an observation sequence of length T with a corresponding label sequence $\mathbf{y} = \langle y_1, \ldots, y_T \rangle$. Words in a sentence correspond to tokens in the input sequence $\mathbf{x}$, which are mapped to labels in $\mathbf{y}$. These labels indicate whether the word belongs to a particular entity class of interest (in this case, org and loc) or not (null). These labels can be assigned by a sequence model based on a finite state machine, such as the one shown to the right in Figure 1.

We focus our discussion of active learning for sequence labeling on *conditional random fields*, or CRFs (Lafferty et al., 2001). The rest of this section serves as a brief introduction. CRFs are statistical graphical models which have demonstrated state-of-the-art accuracy on virtually all of the sequence labeling tasks mentioned in Section 1. We use linear-chain CRFs, which correspond to conditionally trained probabilistic finite state machines.

A linear-chain CRF model with parameters θ defines the posterior probability of $\mathbf{y}$ given $\mathbf{x}$ to be[1]:

$$P(\mathbf{y}|\mathbf{x};\theta) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_{t=1}^{T}\sum_{k=1}^{K} \theta_k f_k(y_{t-1}, y_t, \mathbf{x}_t)\right). \tag{1}$$

[1]Our discussion assumes, without loss of generality, that each label is uniquely represented by one state, thus each label sequence $\mathbf{y}$ corresponds to exactly one path through the model.



x: the ACME Inc. offices in Chicago ...
y: null org org null null loc
null
org
loc

Figure 1: An information extraction example treated as a sequence labeling task. Also shown is a corresponding sequence model represented as a finite state machine.

Here $Z(\mathbf{x})$ is a normalization factor over all possible labelings of $\mathbf{x}$, and θ_k is one of K model parameter weights corresponding to some feature $f_k(y_{t-1}, y_t, \mathbf{x}_t)$. Each feature f_k describes the sequence $\mathbf{x}$ at position t with label y_t, observed along a transition from label states y_{t-1} to y_t in the finite state machine. Consider the example text from Figure 1. Here, f_k might be the feature WORD=*ACME* and have the value $f_k = 1$ along a transition from the null state to the org state (and 0 elsewhere). Other features set to 1 here might be ALLCAPS and NEXTWORD=*Inc.* The weights in θ are set to maximize the conditional log likelihood ℓ of training sequences in the labeled data set $\mathcal{L}$:

$$\ell(\mathcal{L};\theta) = \sum_{l=1}^{L} \log P(\mathbf{y}^{(l)}|\mathbf{x}^{(l)};\theta) - \sum_{k=1}^{K} \frac{\theta_k^2}{2\sigma^2},$$

where L is the size of the labeled set $\mathcal{L}$, and the second term is a Gaussian regularization penalty on $\|\theta\|$ to prevent over-fitting. After training, labels can be predicted for new sequences using the Viterbi algorithm. For more details on CRFs and their training procedures, see Sutton and McCallum (2006).

Note that, while we describe the active learning algorithms in the next section in terms of linear-chain CRFs, they have analogs for other kinds of sequence models, such as hidden Markov models, or HMMs (Rabiner, 1989), probabilistic context-free grammars (Lari and Young, 1990), and general CRFs (Sutton and McCallum, 2006).

3 Active Learning with Sequence Models

In order to select queries, an active learner must have a way of assessing how *informative* each instance is. Let $\mathbf{x}^*$ be the most informative instance according to some query strategy $\phi(\mathbf{x})$, which is a function used to evaluate each instance $\mathbf{x}$ in the unlabeled pool $\mathcal{U}$.

```
Given: Labeled set $\mathcal{L}$, unlabeled pool $\mathcal{U}$, query
       strategy $\phi(\cdot)$, query batch size $B$
repeat
    // learn a model using the current $\mathcal{L}$
    $\theta = \text{train}(\mathcal{L})$ ;
    for $b = 1$ to $B$ do
        // query the most informative instance
        $\mathbf{x}_b^* = \arg\max_{\mathbf{x}\in\mathcal{U}} \phi(\mathbf{x})$ ;
        // move the labeled query from $\mathcal{U}$ to $\mathcal{L}$
        $\mathcal{L} = \mathcal{L} \cup \langle \mathbf{x}_b^*, \text{label}(\mathbf{x}_b^*)\rangle$ ;
        $\mathcal{U} = \mathcal{U} - \mathbf{x}_b^*$ ;
    end
until some stopping criterion ;
```

Algorithm 1: Pool-based active learning.

Algorithm 1 provides a sketch of the generic pool-based active learning scenario.

In the remainder of this section, we describe various query strategy formulations of $\phi(\cdot)$ that have been used for active learning with sequence models. We also point out where we think these approaches may be flawed, and propose several novel query strategies to address these issues.

3.1 Uncertainty Sampling

One of the most common general frameworks for measuring informativeness is *uncertainty sampling* (Lewis and Catlett, 1994), where a learner queries the instance that it is most uncertain how to label. Culotta and McCallum (2005) employ a simple uncertainty-based strategy for sequence models called **least confidence (LC)**:

$$\phi^{LC}(\mathbf{x}) = 1 - P(\mathbf{y}^*|\mathbf{x};\theta).$$

Here, $\mathbf{y}^*$ is the most likely label sequence, i.e., the Viterbi parse. This approach queries the instance for which the current model has the least confidence in its most likely labeling. For CRFs, this confidence can be calculated using the posterior probability given by Equation (1).

Scheffer et al. (2001) propose another uncertainty strategy, which queries the instance with the smallest margin between the posteriors for its two most likely labelings. We call this approach **margin (M)**:

$$\phi^{M}(\mathbf{x}) = -\big(P(\mathbf{y}_1^*|\mathbf{x};\theta) - P(\mathbf{y}_2^*|\mathbf{x};\theta)\big).$$

Here, $\mathbf{y}_1^*$ and $\mathbf{y}_2^*$ are the first and second best label sequences, respectively. These can be efficiently computed using the N-best algorithm (Schwartz and Chow, 1990), a beam-search generalization of Viterbi, with $N = 2$. The minus sign in front is simply to ensure that ϕ^M acts as a maximizer for use with Algorithm 1.

Another uncertainty-based measure of informativeness is *entropy* (Shannon, 1948). For a discrete random variable Y, the entropy is given by $H(Y) = -\sum_i P(y_i)\log P(y_i)$, and represents the information needed to "encode" the distribution of outcomes for Y. As such, is it often thought of as a measure of uncertainty in machine learning. In active learning, we wish to use the entropy of our model's posteriors over its labelings. One way this has been done with probabilistic sequence models is by computing what we call **token entropy (TE)**:

$$\phi^{TE}(\mathbf{x}) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{m=1}^{M} P_\theta(y_t = m)\log P_\theta(y_t = m), \quad (2)$$

where T is the length of $\mathbf{x}$, m ranges over all possible token labels, and $P_\theta(y_t = m)$ is shorthand for the marginal probability that m is the label at position t in the sequence, according to the model. For CRFs and HMMs, these marginals can be efficiently computed using the *forward* and *backward* algorithms (Rabiner, 1989). The summed token entropies have typically been normalized by sequence length T, to avoid simply querying longer sequences (Baldridge and Osborne, 2004; Hwa, 2004). However, we argue that querying long sequences should not be explicitly discouraged, if in fact they contain more information. Thus, we also propose the **total token entropy (TTE)** measure:

$$\phi^{TTE}(\mathbf{x}) = T \times \phi^{TE}(\mathbf{x}).$$

For most sequence labeling tasks, however, it is more appropriate to consider the entropy of the label sequence $\mathbf{y}$ as a whole, rather than some aggregate of individual token entropies. Thus an alternate query strategy is **sequence entropy (SE)**:

$$\phi^{SE}(\mathbf{x}) = -\sum_{\hat{\mathbf{y}}} P(\hat{\mathbf{y}}|\mathbf{x};\theta)\log P(\hat{\mathbf{y}}|\mathbf{x};\theta), \quad (3)$$

where $\hat{\mathbf{y}}$ ranges over all possible label sequences for input sequence $\mathbf{x}$. Note, however, that the number of possible labelings grows exponentially with the length of $\mathbf{x}$. To make this feasible, previous work (Kim et al., 2006) has employed an approximation we call ***N*-best sequence entropy (NSE)**:

$$\phi^{NSE}(\mathbf{x}) = -\sum_{\hat{\mathbf{y}} \in \mathcal{N}} P(\hat{\mathbf{y}}|\mathbf{x};\theta) \log P(\hat{\mathbf{y}}|\mathbf{x};\theta),$$

where $\mathcal{N} = \{\mathbf{y}_1^*, \ldots, \mathbf{y}_N^*\}$, the set of the N most likely parses, and the posteriors are re-normalized (i.e., $Z(\mathbf{x})$ in Equation (1) only ranges over $\mathcal{N}$). For $N = 2$, this approximation is equivalent to ϕ^M, thus N-best sequence entropy can be thought of as a generalization of the margin approach.

Recently, an efficient entropy calculation via dynamic programming was proposed for CRFs in the context of semi-supervised learning (Mann and McCallum, 2007). We use this algorithm to compute the true sequence entropy (3) for active learning in a constant-time factor of Viterbi's complexity. Hwa (2004) employed a similar approach for active learning with probabilistic context-free grammars.

3.2 Query-By-Committee

Another general active learning framework is the *query-by-committee* (QBC) approach (Seung et al., 1992). In this setting, we use a committee of models $\mathcal{C} = \{\theta^{(1)}, \ldots, \theta^{(C)}\}$ to represent C different hypotheses that are consistent with the labeled set $\mathcal{L}$. The most informative query, then, is the instance over which the committee is in most disagreement about how to label.

In particular, we use the *query-by-bagging* approach (Abe and Mamitsuka, 1998) to learn a committee of CRFs. In each round of active learning, $\mathcal{L}$ is sampled (with replacement) L times to create a unique, modified labeled set $\mathcal{L}^{(c)}$. Each model $\theta^{(c)} \in \mathcal{C}$ is then trained using its own corresponding labeled set $\mathcal{L}^{(c)}$. To measure disagreement among committee members, we consider two alternatives.

Dagan and Engelson (1995) introduced QBC with HMMs for part-of-speech tagging using a measure called **vote entropy (VE)**:

$$\phi^{VE}(\mathbf{x}) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{m=1}^{M} \frac{V(y_t,m)}{C} \log \frac{V(y_t,m)}{C},$$

where $V(y_t, m)$ is the number of "votes" label m receives from all the committee member's Viterbi labelings at sequence position t.

McCallum and Nigam (1998) propose a QBC strategy for classification based on *Kullback-Leibler (KL) divergence*, an information-theoretic measure of the difference between two probability distributions. The most informative query is considered to be the one with the largest average KL divergence between a committee member's posterior label distribution and the consensus. We modify this approach for sequence models by summing the average KL scores using the marginals at each token position and, as with vote entropy, normalizing for length. We call this approach **Kullback-Leibler (KL)**:

$$\phi^{KL}(\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} \frac{1}{C}\sum_{c=1}^{C} D(\theta^{(c)} \| \mathcal{C}),$$

where (using shorthand again):

$$D(\theta^{(c)} \| \mathcal{C}) = \sum_{m=1}^{M} P_{\theta^{(c)}}(y_t = m) \log \frac{P_{\theta^{(c)}}(y_t = m)}{P_{\mathcal{C}}(y_t = m)}.$$

Here $P_{\mathcal{C}}(y_t = m) = \frac{1}{C}\sum_{c=1}^{C} P_{\theta^{(c)}}(y_t = m)$, or the "consensus" marginal probability that m is the label at position t in the sequence.

Both of these disagreement measures are normalized for sequence length T. As with token entropy (2), this may bias the learner toward querying shorter sequences. To study the effects of normalization, we also conduct experiments with **non-normalized variants** ϕ^{TVE} and ϕ^{TKL}.

Additionally, we argue that these token-level disagreement measures may be less appropriate for most tasks than measuring the committee's disagreement about the label sequence $\mathbf{y}$ as a whole. Therefore, we propose **sequence vote entropy (SVE)**:

$$\phi^{SVE}(\mathbf{x}) = -\sum_{\hat{\mathbf{y}} \in \mathcal{N}^{\mathcal{C}}} P(\hat{\mathbf{y}}|\mathbf{x};\mathcal{C}) \log P(\hat{\mathbf{y}}|\mathbf{x};\mathcal{C}),$$

where $\mathcal{N}^{\mathcal{C}}$ is the union of the N-best parses from all models in the committee $\mathcal{C}$, and $P(\hat{\mathbf{y}}|\mathbf{x};\mathcal{C}) = \frac{1}{C}\sum_{c=1}^{C} P(\hat{\mathbf{y}}|\mathbf{x};\theta^{(c)})$, or the "consensus" posterior probability for some label sequence $\hat{\mathbf{y}}$. This can be thought of as a QBC generalization of N-best entropy, where each committee member casts a vote for the posterior label distribution. We also explore a **sequence Kullback-Leibler (SKL)** variant:

$$\phi^{SKL}(\mathbf{x}) = \frac{1}{C}\sum_{c=1}^{C}\sum_{\hat{\mathbf{y}} \in \mathcal{N}^{\mathcal{C}}} P(\hat{\mathbf{y}}|\mathbf{x};\theta^{(c)}) \log \frac{P(\hat{\mathbf{y}}|\mathbf{x};\theta^{(c)})}{P(\hat{\mathbf{y}}|\mathbf{x};\mathcal{C})}.$$

3.3 Expected Gradient Length

A third general active learning framework we consider is to query the instance that would impart the greatest change to the current model *if we knew its label*. Since we train discriminative models like CRFs using gradient-based optimization, this involves querying the instance which, if labeled and added to the training set, would create the greatest change in the gradient of the objective function (i.e., the largest gradient vector used to re-estimate parameter values).

Let $\nabla\ell(\mathcal{L};\theta)$ be the gradient of the log-likelihood ℓ with respect to the model parameters θ, as given by Sutton and McCallum (2006). Now let $\nabla\ell(\mathcal{L}^{+\langle\mathbf{x},\mathbf{y}\rangle};\theta)$ be the new gradient that would be obtained by adding the training tuple $\langle\mathbf{x},\mathbf{y}\rangle$ to $\mathcal{L}$. Since the query algorithm does not know the true label sequence $\mathbf{y}$ in advance, we instead calculate the **expected gradient length (EGL)**:

$$\phi^{EGL}(\mathbf{x}) = \sum_{\hat{\mathbf{y}}\in\mathcal{N}} P(\hat{\mathbf{y}}|\mathbf{x};\theta)\left\|\nabla\ell(\mathcal{L}^{+\langle\mathbf{x},\hat{\mathbf{y}}\rangle};\theta)\right\|,$$

approximated as an expectation over the N-best labelings, where $\|\cdot\|$ is the Euclidean norm of each resulting gradient vector. We first introduced this approach in previous work on multiple-instance active learning (Settles et al., 2008), and adapt it to query selection with sequences here. Note that, at query time, $\nabla\ell(\mathcal{L};\theta)$ should be nearly zero since ℓ converged at the previous round of training. Thus, we can approximate $\nabla\ell(\mathcal{L}^{+\langle\mathbf{x},\hat{\mathbf{y}}\rangle};\theta) \approx \nabla\ell(\langle\mathbf{x},\hat{\mathbf{y}}\rangle;\theta)$ for computational efficiency, because the training instances are assumed to be independent.

3.4 Information Density

It has been suggested that uncertainty sampling and QBC are prone to querying outliers (Roy and McCallum, 2001; Zhu et al., 2003). Figure 2 illustrates this problem for a binary linear classifier using uncertainty sampling. The least certain instance lies on the classification boundary, but is not "representative" of other instances in the distribution, so knowing its label is unlikely to improve accuracy on the data as a whole. QBC and EGL exhibit similar behavior, by spending time querying possible outliers simply because they are controversial, or are expected to impart significant change in the model.

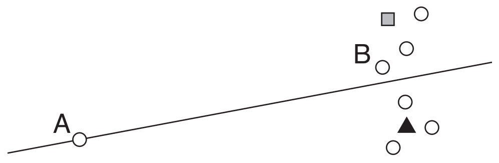


Figure 2: An illustration of when uncertainty sampling can be a poor strategy for classification. Shaded polygons represent labeled instances ($\mathcal{L}$), and circles represent unlabeled instances ($\mathcal{U}$). Since A is on the decision boundary, it will be queried as the most uncertain. However, querying B is likely to result in more information about the data as a whole.

We argue that this phenomenon can occur with sequence labeling tasks as well as with classification. To address this, we propose a new active learning approach called **information density (ID)**:

$$\phi^{ID}(\mathbf{x}) = \phi^{SE}(\mathbf{x}) \times \left(\frac{1}{U}\sum_{u=1}^{U}\text{sim}(\mathbf{x},\mathbf{x}^{(u)})\right)^{\beta}.$$

That is, the informativeness of $\mathbf{x}$ is weighted by its average similarity to all other sequences in $\mathcal{U}$, subject to a parameter β that controls the relative importance of the density term. In the formulation presented above, sequence entropy ϕ^{SE} measures the "base" informativeness, but we could just as easily use any of the instance-level strategies presented in the previous sections.

This density measure requires us to compute the similarity of two sequences. To do this, we first transform each $\mathbf{x}$, which is a sequence of feature vectors (tokens), into a single kernel vector $\vec{\mathbf{x}}$:

$$\vec{\mathbf{x}} = \left[\sum_{t=1}^{T} f_1(x_t), \ldots, \sum_{t=1}^{T} f_J(x_t)\right],$$

where $f_j(x_t)$ is the value of feature f_j for token x_t, and J is the number of features in the input representation[2]. In other words, sequence $\mathbf{x}$ is compressed into a fixed-length feature vector $\vec{\mathbf{x}}$, for which each element is the sum of the corresponding feature's values across all tokens. We can then use cosine

[2]Note that $J \neq K$, and $f_j(x_t)$ here differs slightly from the feature definition given in Section 2. Since the labels y_{t-1} and y_t are unknown before querying, the K features used for model training are reduced down to the J input features here, which factor out any label dependencies.

similarity on this simplified representation:

$$\text{sim}_{cos}(\mathbf{x}, \mathbf{x}^{(u)}) = \frac{\vec{\mathbf{x}} \cdot \vec{\mathbf{x}}^{(u)}}{\|\vec{\mathbf{x}}\| \times \|\vec{\mathbf{x}}^{(u)}\|}.$$

We have also investigated similarity functions based on exponentiated Euclidean distance and KL-divergence, the latter of which was also employed by McCallum and Nigam (1998) for density-weighting QBC in text classification. However, these measures show no improvement over cosine similarity, and require setting additional hyper-parameters.

One potential drawback of information density is that the number of required similarity calculations grows quadratically with the number of instances in $\mathcal{U}$. For pool-based active learning, we often assume that the size of $\mathcal{U}$ is very large. However, these densities only need to be computed once, and are independent of the base information measure. Thus, when employing information density in a real-world interactive learning setting, the density scores can simply be pre-computed and cached for efficient lookup during the actual active learning process.

3.5 Fisher Information

We also introduce a query selection strategy for sequence models based on *Fisher information*, building on the theoretical framework of Zhang and Oles (2000). Fisher information $\mathcal{I}(\theta)$ represents the overall uncertainty about the estimated model parameters θ, as given by:

$$\mathcal{I}(\theta) = -\int_{\mathbf{x}} P(\mathbf{x}) \int_{\mathbf{y}} P(\mathbf{y}|\mathbf{x};\theta) \frac{\partial^2}{\partial\theta^2} \log P(\mathbf{y}|\mathbf{x};\theta).$$

For a model with K parameters, the Fisher information takes the form of a $K \times K$ covariance matrix. Our goal in active learning is to select the query that most efficiently minimizes the model variance reflected in $\mathcal{I}(\theta)$. This can be accomplished by optimizing the **Fisher information ratio (FIR)**:

$$\phi^{FIR}(\mathbf{x}) = -\text{tr}\left(\mathcal{I}_{\mathbf{x}}(\theta)^{-1}\mathcal{I}_{\mathcal{U}}(\theta)\right), \qquad (4)$$

where $\mathcal{I}_{\mathbf{x}}(\theta)$ and $\mathcal{I}_{\mathcal{U}}(\theta)$ are Fisher information matrices for sequence $\mathbf{x}$ and the unlabeled pool $\mathcal{U}$, respectively. The leading minus sign again ensures that ϕ^{FIR} is a maximizer for use with Algorithm 1.

Previously, Fisher information for active learning has only been investigated in the context of simple binary classification. When employing FIR with sequence models like CRFs, there are two additional computational challenges. First, we must integrate over all possible labelings $\mathbf{y}$, which can, as we have seen, be approximated as an expectation over the N-best labelings. Second, the inner product in the ratio calculation (4) requires inverting a $K \times K$ matrix for each $\mathbf{x}$. In most interesting natural language applications, K is very large, making this algorithm intractable. However, it is common in similar situations to approximate the Fisher information matrix with its diagonal (Nyffenegger et al., 2006). Thus we estimate $\mathcal{I}_{\mathbf{x}}(\theta)$ using:

$$\mathcal{I}_{\mathbf{x}}(\theta) = \sum_{\hat{\mathbf{y}} \in \mathcal{N}} P(\hat{\mathbf{y}}|\mathbf{x};\theta) \Bigg[\left(\frac{\partial \log P(\hat{\mathbf{y}}|\mathbf{x};\theta)}{\partial \theta_1} \right)^2 + \delta, \ldots, \left(\frac{\partial \log P(\hat{\mathbf{y}}|\mathbf{x};\theta)}{\partial \theta_K} \right)^2 + \delta \Bigg],$$

and $\mathcal{I}_{\mathcal{U}}(\theta)$ using:

$$\mathcal{I}_{\mathcal{U}}(\theta) = \frac{1}{U} \sum_{u=1}^{U} \mathcal{I}_{\mathbf{x}^{(u)}}(\theta).$$

For CRFs, the partial derivative at the root of each element in the diagonal vector is given by:

$$\frac{\partial \log P(\hat{\mathbf{y}}|\mathbf{x};\theta)}{\partial \theta_k} = \sum_{t=1}^{T} f_k(\hat{y}_{t-1}, \hat{y}_t, \mathbf{x}_t) - \sum_{t=1}^{T} \sum_{y,y'} P(y, y'|\mathbf{x}) f_k(y, y', \mathbf{x}_t),$$

which is similar to the equation used to compute the training gradient, but without a regularization term. A smoothing parameter $\delta \ll 1$ is added to prevent division by zero when computing the ratio.

Notice that this method implicitly selects representative instances by favoring queries with Fisher information $\mathcal{I}_{\mathbf{x}}(\theta)$ that is not only high, but similar to that of the overall data distribution $\mathcal{I}_{\mathcal{U}}(\theta)$. This is in contrast to information density, which tries to query representative instances by explicitly modeling the distribution with a density weight.

Corpus	Entities	Features	Instances
CoNLL-03	4	78,644	19,959
NLPBA	5	128,401	18,854
BioCreative	1	175,331	10,000
FlySlip	1	31,353	1,220
CORA:Headers	15	22,077	935
CORA:References	13	4,208	500
Sig+Reply	2	25	617
SigIE	12	10,600	250

Table 1: Properties of the different evaluation corpora.

4 Empirical Evaluation

In this section we present a large-scale empirical analysis of the query strategies described in Section 3 on eight benchmark information extraction and document segmentation corpora. The data sets are summarized in Table 1.

4.1 Data and Methodology

CoNLL-03 (Sang and DeMeulder, 2003) is a collection of newswire articles annotated with four entities: person, organization, location, and misc. NLPBA (Kim et al., 2004) is a large collection of biomedical abstracts annotated with five entities of interest, such as protein, RNA, and cell-type. BioCreative (Yeh et al., 2005) and FlySlip (Vlachos, 2007) also comprise texts in the biomedical domain, annotated for gene entity mentions in articles from the human and fruit fly literature, respectively. CORA (Peng and McCallum, 2004) consists of two collections: a set of research paper headers annotated for entities such as title, author, and institution; and a collection of references annotated with BibTeX fields such as journal, year, and publisher. The Sig+Reply corpus (Carvalho and Cohen, 2004) is a set of email messages annotated for signature and quoted reply line segments. SigIE is a subset of the signature blocks from Sig+Reply which we have enhanced with several address book fields such as name, email, and phone. All corpora are formatted in the "IOB" sequence representation (Ramshaw and Marcus, 1995).

We implement all fifteen query selection strategies described in Section 3 for use with CRFs, and evaluate them on all eight data sets. We also compare against two baseline strategies: random instance selection (i.e., passive learning), and naïvely querying the longest sequence in terms of tokens. We use a typical feature set for each corpus based on the cited literature (including words, orthographic patterns, part-of-speech, lexicons, etc.). Where the N-best approximation is used $N = 15$, and for all QBC methods $C = 3$; these figures exhibited a good balance of accuracy and training speed in preliminary work. For information density, we arbitrarily set $\beta = 1$ (i.e., the information and density terms have equal weight). In each experiment, $\mathcal{L}$ is initialized with five random labeled instances, and up to 150 queries are subsequently selected from $\mathcal{U}$ in batches of size $B = 5$. All results are averaged across five folds using cross-validation.

We evaluate each query strategy by constructing learning curves that plot the overall F_1 measure (for all entities or segments) as a function of the number of instances queried. Due to lack of space, we cannot show learning curves for every experiment. Instead, Table 2 summarizes our results by reporting the area under the learning curve for all strategies on all data. Figure 3 presents a few representative learning curves for six of the corpora.

4.2 Discussion of Learning Curves

The first conclusion we can draw from these results is that there is no single clear winner. However, information density (ID), which we introduce in this paper, stands out. It usually improves upon the base sequence entropy measure, never performs poorly, and has the highest average area under the learning curve across all tasks. It seems particularly effective on large corpora, which is a typical assumption for the active learning setting. Sequence vote entropy (SVE), a QBC method we propose here, is also noteworthy in that it is fairly consistently among the top three strategies, although never the best.

Second, the top uncertainty sampling strategies are least confidence (LC) and sequence entropy (SE), the latter being the dominant entropy-based method. Among the QBC strategies, sequence vote entropy (SVE) is the clear winner. We conclude that these three methods are the best base information measures for use with information density.

Third, query strategies that evaluate the entire sequence (SE, SVE, SKL) are generally superior to those which aggregate token-level information. Furthermore, the *total* token-level strategies (TTE, TVE, TKL) outperform their *length-*

	Baselines		**Uncertainty Sampling**						**Query-By-Committee**						**Other**		
Corpus	**Rand**	**Long**	**LC**	**M**	**TE**	***TTE***	**SE**	**NSE**	**VE**	***KL***	***TVE***	***TKL***	***SVE***	***SKL***	***EGL***	***ID***	***FIR***
CoNLL-03	78.8	79.4	89.4	84.5	38.9	$\underline{\textbf{89.7}}$	$\boxed{\textbf{90.1}}$	89.1	45.9	62.0	86.7	81.7	89.0	87.9	87.3	**89.6**	81.7
NLPBA	59.9	67.6	71.0	62.9	53.4	70.9	71.5	68.9	52.4	53.1	66.9	63.5	**71.8**	68.5	69.3	$\underline{\textbf{73.1}}$	$\boxed{\textbf{73.6}}$
BioCreative	34.6	26.9	54.8	46.8	37.8	53.0	56.0	50.5	35.2	37.4	49.2	45.1	**56.6**	50.8	51.5	$\boxed{\textbf{59.1}}$	$\underline{\textbf{58.8}}$
FlySlip	112.1	121.0	125.1	119.5	110.3	124.9	**125.4**	124.1	113.3	109.4	124.1	119.5	122.7	120.7	$\underline{\textbf{125.9}}$	$\boxed{\textbf{126.8}}$	118.2
Headers	76.0	78.2	$\boxed{\textbf{81.4}}$	78.6	78.5	78.5	$\underline{\textbf{80.8}}$	80.4	72.8	78.5	79.7	78.5	**80.7**	78.4	79.6	80.2	79.1
References	$\underline{\textbf{90.0}}$	86.0	89.8	$\boxed{\textbf{91.5}}$	84.4	88.6	88.4	89.4	85.1	89.1	88.7	88.2	**89.9**	86.9	88.2	88.7	87.1
Sig+Reply	129.1	129.6	132.1	132.3	131.7	131.6	131.4	$\underline{\textbf{133.1}}$	131.4	130.7	132.1	130.6	**132.8**	132.3	130.5	131.5	$\boxed{\textbf{133.2}}$
SigIE	84.3	82.7	88.8	87.3	89.3	88.3	87.6	89.1	$\boxed{\textbf{89.8}}$	85.5	$\underline{\textbf{89.7}}$	85.1	**89.5**	$\underline{\textbf{89.7}}$	87.7	88.5	88.5
Average	83.1	83.9	$\underline{\textbf{91.6}}$	87.9	78.0	90.7	**91.4**	90.6	78.2	80.7	89.6	86.5	$\underline{\textbf{91.6}}$	89.4	90.0	$\boxed{\textbf{92.2}}$	90.0

Table 2: Detailed results for all query strategies on all evaluation corpora. Reported is the area under the F_1 learning curve for each strategy after 150 queries (maximum possible score is 150). For each row, the $\boxed{\textbf{best method}}$ is shown boxed in bold, the $\underline{\textbf{second best}}$ is shown underlined in bold, and the **third best** is shown in bold. The last row summarizes the results across all eight tasks by reporting the average area for each strategy. Query strategy formulations for sequence models introduced in this paper are indicated with italics along the top.

normalized counterparts (TE, VE, KL) in nearly all cases. In fact, the normalized variants are often inferior even to the baselines. While an argument can be made that these shorter sequences might be easier to label from a human annotator's perspective, our ongoing work indicates that the relationship between instance length and actual labeling costs (e.g., elapsed annotation time) is not a simple one. Analysis of our experiment logs also shows that length-normalized methods are occasionally biased toward short sequences with little intuitive value (e.g., sentences with few or no entities to label). In addition, vote entropy appears to be a better disagreement measure for QBC strategies than KL divergence.

Finally, Fisher information (FIR), while theoretically sound, exhibits behavior that is difficult to interpret. It is sometimes the winning strategy, but occasionally only on par with the baselines. When it does show significant gains over the other strategies, these gains appear to be only for the first several queries (e.g., NLPBA and BioCreative in Figure 3). This inconsistent performance may be a result of the approximations made for computational efficiency. Expected gradient length (EGL) also appears to exhibit mediocre performance, and is likely not worth its additional computational expense.

4.3 Discussion of Run Times

Here we discuss the execution times for each query strategy using current hardware. The uncertainty sampling methods are roughly comparable in run time (token-based methods run slightly faster), each routinely evaluating tens of thousands of sequences in under a minute. The QBC methods, on the other hand, must re-train multiple models with each query, resulting in a lag of three to four minutes per query batch (and up to 20 minutes for corpora with more entity labels).

The expected gradient length and Fisher information methods are the most computationally expensive, because they must first perform inference over the possible labelings and then calculate gradients for each candidate label sequence. As a result, they take eight to ten minutes (upwards of a half hour on the larger corpora) for each query. Unlike the other strategies, their time complexities also scale linearly with the number of model parameters K which, in turn, increases as new sequences are added to $\mathcal{L}$.

As noted in Section 3.4, information density incurs a large computational cost to estimate the density weights, but these can be pre-computed and cached for efficient lookup. In our experiments, this pre-processing step takes less than a minute for the smaller corpora, about a half hour for CoNLL-03 and BioCreative, and under two hours for NLPBA. The density lookup causes no significant change in the run time of the base information measure. Given these results, we advocate information density with an uncertainty sampling base measure in practice, particularly for active learning with large corpora.

5 Conclusion

In this paper, we have presented a detailed analysis of active learning for sequence labeling tasks. In particular, we have described and criticized the query selection strategies used with probabilistic se-

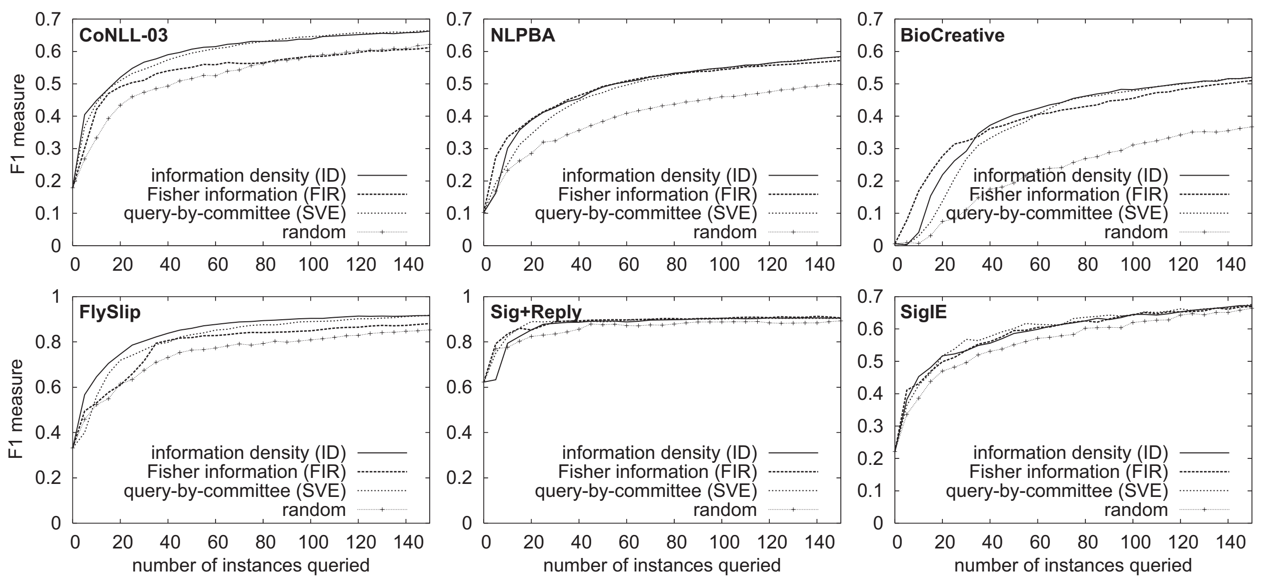


Figure 3: Learning curves for selected query strategies on six of the evaluation corpora.

quence models to date, and proposed several novel strategies to address some of their shortcomings. Our large-scale empirical evaluation demonstrates that some of these newly proposed methods advance the state of the art in active learning with sequence models. These methods include information density (which we recommend in practice), sequence vote entropy, and sometimes Fisher information.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful feedback. This work was supported by NIH grants T15-LM07359 and R01-LM07050.

References

N. Abe and H. Mamitsuka. 1998. Query learning strategies using boosting and bagging. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1–9. Morgan Kaufmann.

J. Baldridge and M. Osborne. 2004. Active learning and the total cost of annotation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9–16. ACL Press.

V.R. Carvalho and W. Cohen. 2004. Learning to extract signature and reply lines from email. In *Proceedings of the Conference on Email and Anti-Spam (CEAS)*.

D. Cohn, L. Atlas, and R. Ladner. 1994. Improving generalization with active learning. *Machine Learning*, 15(2):201–221.

A. Culotta and A. McCallum. 2005. Reducing labeling effort for stuctured prediction tasks. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 746–751. AAAI Press.

I. Dagan and S. Engelson. 1995. Committee-based sampling for training probabilistic classifiers. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 150–157. Morgan Kaufmann.

R. Hwa. 2004. Sample selection for statistical parsing. *Computational Linguistics*, 30(3):73–77.

J. Kim, T. Ohta, Y. Tsuruoka, Y. Tateisi, and N. Collier. 2004. Introduction to the bio-entity recognition task at JNLPBA. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA)*, pages 70–75.

S. Kim, Y. Song, K. Kim, J.W. Cha, and G.G. Lee. 2006. MMR-based active machine learning for bio named entity recognition. In *Proceedings of Human Language Technology and the North American Association for Computational Linguistics (HLT-NAACL)*, pages 69–72. ACL Press.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 282–289. Morgan Kaufmann.

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.

D. Lewis and J. Catlett. 1994. Heterogeneous uncertainty sampling for supervised learning. In *Pro-*

ceedings of the International Conference on Machine Learning (ICML), pages 148–156. Morgan Kaufmann.

G. Mann and A. McCallum. 2007. Efficient computation of entropy gradient for semi-supervised conditional random fields. In *Proceedings of the North American Association for Computational Linguistics (NAACL)*, pages 109–112. ACL Press.

A. McCallum and K. Nigam. 1998. Employing EM in pool-based active learning for text classification. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 359–367. Morgan Kaufmann.

M. Nyffenegger, J.C. Chappelier, and E. Gaussier. 2006. Revisiting Fisher kernels for document similarities. In *Proceedings of the European Conference on Machine Learning (ECML)*, pages 727–734. Springer.

F. Peng and A. McCallum. 2004. Accurate information extraction from research papers using conditional random fields. In *Proceedings of Human Language Technology and the North American Association for Computational Linguistics (HLT-NAACL)*. ACL Press.

L. R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

L.A. Ramshaw and M.P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the ACL Workshop on Very Large Corpora*.

N. Roy and A. McCallum. 2001. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 441–448. Morgan Kaufmann.

E.F.T.K. Sang and F. DeMeulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*, pages 142–147.

T. Scheffer, C. Decomain, and S. Wrobel. 2001. Active hidden Markov models for information extraction. In *Proceedings of the International Conference on Advances in Intelligent Data Analysis (CAIDA)*, pages 309–318. Springer-Verlag.

R. Schwartz and Y.-L. Chow. 1990. The N-best algorithm: an efficient and exact procedure for finding the N most likely sentence hypotheses. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 81–83. IEEE Press.

B. Settles, M. Craven, and S. Ray. 2008. Multiple-instance active learning. In *Advances in Neural Information Processing Systems (NIPS)*, volume 20, pages 1289–1296. MIT Press.

H.S. Seung, M. Opper, and H. Sompolinsky. 1992. Query by committee. In *Proceedings of the ACM Workshop on Computational Learning Theory*, pages 287–294.

C. E. Shannon. 1948. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423,623–656.

C. Sutton and A. McCallum. 2006. An introduction to conditional random fields for relational learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press.

A. Vlachos. 2007. Evaluating and combining biomedical named entity recognition systems. In *BioNLP 2007: Biological, translational, and clinical language processing*, pages 199–206.

A. Yeh, A. Morgan, M. Colosimo, and L. Hirschman. 2005. Biocreative task 1a: gene mention finding evaluation. *BMC Bioinformatics*, 6(Suppl 1):S2.

T. Zhang and F.J. Oles. 2000. A probability analysis on the value of unlabeled data for classification problems. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1191–1198. Morgan Kaufmann.

X. Zhu, J. Lafferty, and Z. Ghahramani. 2003. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the ICML Workshop on the Continuum from Labeled to Unlabeled Data*, pages 58–65.

Latent-Variable Modeling of String Transductions with Finite-State Methods[*]

Markus Dreyer and **Jason R. Smith** and **Jason Eisner**
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218, USA
{markus,jsmith,jason}@cs.jhu.edu

Abstract

String-to-string transduction is a central problem in computational linguistics and natural language processing. It occurs in tasks as diverse as name transliteration, spelling correction, pronunciation modeling and inflectional morphology. We present a conditional log-linear model for string-to-string transduction, which employs overlapping features over latent alignment sequences, and which learns latent classes and latent string pair regions from incomplete training data. We evaluate our approach on morphological tasks and demonstrate that latent variables can dramatically improve results, even when trained on small data sets. On the task of generating morphological forms, we outperform a baseline method reducing the error rate by up to 48%. On a lemmatization task, we reduce the error rates in Wicentowski (2002) by 38–92%.

1 Introduction

A recurring problem in computational linguistics and language processing is transduction of character strings, e.g., words. That is, one wishes to model some systematic mapping from an input string x to an output string y. Applications include:

- *phonology*: underlying representation $\leftrightarrow$ surface representation
- *orthography*: pronunciation $\leftrightarrow$ spelling
- *morphology*: inflected form $\leftrightarrow$ lemma, or differently inflected form
- *fuzzy name matching* (duplicate detection) and *spelling correction*: spelling $\leftrightarrow$ variant spelling
- *lexical translation* (cognates, loanwords, transliterated names): English word $\leftrightarrow$ foreign word

We present a configurable and robust framework for solving such word transduction problems. Our results in morphology generation show that the presented approach improves upon the state of the art.

2 Model Structure

A weighted edit distance model (Ristad and Yianilos, 1998) would consider each character in isolation. To consider more context, we pursue a very natural generalization. Given an input x, we evaluate a candidate output y by moving a sliding window over the aligned (x, y) pair. More precisely, since many alignments are possible, we sum over all these possibilities, evaluating each alignment *separately*.[1]

At each window position, we accumulate log-probability based on the material that appears within the current window. The window is a few characters wide, and successive window positions *overlap*. This stands in contrast to a competing approach (Sherif and Kondrak, 2007; Zhao et al., 2007) that is inspired by phrase-based machine translation (Koehn et al., 2007), which *segments* the input string into substrings that are transduced *independently*, ignoring context.[2]

[*]This work was supported by the Human Language Technology Center of Excellence and by National Science Foundation grant No. 0347822 to the final author. We would also like to thank Richard Wicentowski for providing us with datasets for lemmatization, and the anonymous reviewers for their valuable feedback.

[1]At the other extreme, Freitag and Khadivi (2007) use no alignment; each feature takes its own view of how (x, y) relate.

[2]We feel that this independence is inappropriate. By analogy, it would be a poor idea for a language model to score a string highly if it could be *segmented* into independently frequent n-grams. Rather, language models use overlapping n-grams (indeed, it is the language model that rescues phrase-based MT from producing disjointed translations). We believe phrase-based MT avoids overlapping phrases in the *channel* model only because these would complicate the modeling of reordering (though see, e.g., Schwenk et al. (2007) and Casacuberta (2000)). But in the problems of section 1, letter reordering is rare and we may assume it is local to a window.

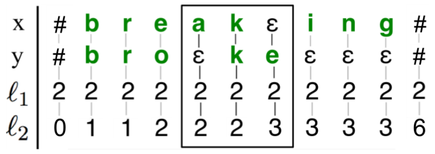


Figure 1: One of many possible alignment strings A for the observed pair *breaking/broke*, enriched with latent strings ℓ_1 and ℓ_2. Observed letters are shown in bold. The box marks a trigram to be scored. See Fig. 2 for features that fire on this trigram.

Joint n-gram models over the input and output dimensions have been used before, but not for morphology, where we will apply them.[3] Most notable is the local log-linear grapheme-to-phoneme model of Chen (2003), as well as generative models for that task (Deligne et al. (1995), Galescu and Allen (2001), Bisani and Ney (2002)).

We advance that approach by adding new *latent* dimensions to the (input, output) tuples (see Fig. 1).[4] This enables us to use certain linguistically inspired features and discover unannotated information. Our features consider *less* or *more* than a literal n-gram. On the one hand, we generalize with features that abstract away from the n-gram window contents; on the other, we specialize the n-gram with features that make use of the added latent linguistic structure.

In section 5, we briefly sketch our framework for concisely expressing and efficiently implementing models of this form. Our framework uses familiar log-linear techniques for stochastic modeling, and weighted finite-state methods both for implementation and for specifying features. It appears general enough to cover most prior work on word transduction. We imagine that it will be useful for future work as well: one might easily add new, linguistically interesting classes of features, each class defined by a regular expression.

2.1 Basic notation

We use an **input alphabet** Σ_x and **output alphabet** Σ_y. We conventionally use $x \in \Sigma_\text{x}^*$ to denote the input string and $y \in \Sigma_\text{y}^*$ to denote the output string.

[3]Clark (2001) does use pair HMMs for morphology.

[4]Demberg et al. (2007) similarly added extra dimensions. However, their added dimensions were supervised, not latent, and their model was a standard generative n-gram model whose generalization was limited to standard n-gram smoothing.

There are many possible alignments between x and y. We represent each as an **alignment string** $A \in \Sigma_\text{xy}^*$, over an **alignment alphabet** of ordered pairs, $\Sigma_\text{xy} \overset{\text{def}}{=} ((\Sigma_\text{x} \cup \{\epsilon\}) \times (\Sigma_\text{y} \cup \{\epsilon\})) - \{(\epsilon, \epsilon)\}$.

For example, one alignment of $x = \texttt{breaking}$ with $y = \texttt{broke}$ is the 9-character string $A = (\texttt{b},\texttt{b})(\texttt{r},\texttt{r})(\texttt{e},\texttt{o})(\texttt{a},\epsilon)(\texttt{k},\texttt{k})(\epsilon,\texttt{e})(\texttt{i},\epsilon)(\texttt{n},\epsilon)(\texttt{g},\epsilon)$. It is pictured in the first two lines of Fig. 1.

The remainder of Fig. 1 shows how we introduce latent variables, by enriching the alignment characters to be tuples rather than pairs. Let $\Sigma \overset{\text{def}}{=} (\Sigma_\text{xy} \times \Sigma_{\ell_1} \times \Sigma_{\ell_2} \times \cdots \times \Sigma_{\ell_K})$, where Σ_{ℓ_i} are alphabets used for the latent variables ℓ_i.

FSA and **FST** stand for "finite-state acceptor" and "finite-state transducer," while **WFSA** and **WFST** are their weighted variants. The $\circ$ symbol denotes composition.

Let T be a relation and w a string. We write $T[w]$ to denote the image of w under T (i.e., range($w \circ T$)), a set of 0 or more strings. Similarly, if W is a weighted language (typically encoded by a WFSA), we write $W[w]$ to denote the weight of w in W.

Let $\pi_\text{x} \subseteq \Sigma^* \times \Sigma_\text{x}^*$ denote the deterministic regular relation that projects an alignment string to its corresponding input string, so that $\pi_\text{x}[A] = x$. Similarly, define $\pi_\text{y} \subseteq \Sigma^* \times \Sigma_\text{y}^*$ so that $\pi_\text{y}[A] = y$. Let $\boldsymbol{A_{xy}}$ be the set of alignment strings A compatible with x and y; formally, $\boldsymbol{A_{xy}} \overset{\text{def}}{=} \{A \in \Sigma^* : \pi_\text{x}[A] = x \wedge \pi_\text{y}[A] = y\}$. This set will range over all possible alignments between x and y, and *also* all possible configurations of the latent variables.

2.2 Log-linear modeling

We use a standard log-linear model whose features are defined on alignment strings $A \in \boldsymbol{A_{xy}}$, allowing them to be sensitive to the alignment of x and y. Given a collection of features $f_i : \Sigma^* \to \mathbb{R}$ with associated weights $\theta_i \in \mathbb{R}$, the conditional likelihood of the training data is

$$p_{\boldsymbol{\theta}}(y \mid x) = \frac{\sum_{A \in \boldsymbol{A_{xy}}} \exp \sum_i \theta_i f_i(A)}{\sum_{y'} \sum_{A \in \boldsymbol{A_{xy'}}} \exp \sum_i \theta_i f_i(A)} \quad (1)$$

Given a parameter vector $\boldsymbol{\theta}$, we compute equation (1) using a finite-state machine. We define a WFSA, $U_{\boldsymbol{\theta}}$, such that $U_{\boldsymbol{\theta}}[A]$ yields the unnormalized probability $u_{\boldsymbol{\theta}}(A) \overset{\text{def}}{=} \exp \sum_i \theta_i f_i(A)$ for any $A \in \Sigma^*$. (See section 5 for the construction.) To obtain the numerator of equation (1), with its $\sum_{A \in \boldsymbol{A}_{\boldsymbol{xy}}}$, we sum over all paths in $U_{\boldsymbol{\theta}}$ that are compatible with x and y. That is, we build $x \circ \pi_{\mathrm{x}}^{-1} \circ U_{\boldsymbol{\theta}} \circ \pi_{\mathrm{y}} \circ y$ and sum over all paths. For the denominator we build the larger machine $x \circ \pi_{\mathrm{x}}^{-1} \circ U_{\boldsymbol{\theta}}$ and again compute the pathsum. We use standard algorithms (Eisner, 2002) to compute the pathsums as well as their gradients with respect to $\boldsymbol{\theta}$ for optimization (section 4.1).

Below, we will restrict our notion of *valid* alignment strings in Σ^*. $U_{\boldsymbol{\theta}}$ is constructed not to accept *invalid* ones, thus assigning them probability 0.

Note that the possible output strings y' in the denominator in equation (1) may have arbitrary length, leading to an infinite summation over alignment strings. Thus, for some values of $\boldsymbol{\theta}$, the sum in the denominator diverges and the probability distribution is undefined. There exist principled ways to avoid such $\boldsymbol{\theta}$ during training. However, in our current work, we simply restrict to finitely many alignment strings (given x), by prohibiting as invalid those with $> k$ *consecutive* insertions (i.e., characters like $(\epsilon, \mathtt{a})$).[5] Finkel et al. (2008) and others have similarly bounded unary rule cycles in PCFGs.

2.3 Latent variables

The alignment between x and y is a latent explanatory variable that helps model the distribution $p(y \mid x)$ but is not observed in training. Other latent variables can also be useful. Morphophonological changes are often sensitive to *phonemes* (whereas x and y may consist of graphemes); *syllable boundaries*; a *conjugation class*; *morpheme boundaries*; and the *position* of the change within the form.

Thus, as mentioned in section 2.1, we enrich the alignment string A so that it specifies additional latent variables to which features may wish to refer. In Fig. 1, two latent strings are added, enabling the features in Fig. 2(a)–(h). The first character is not just an input/output pair, but the 4-tuple $(\mathtt{b}, \mathtt{b}, \mathtt{2}, \mathtt{1})$.

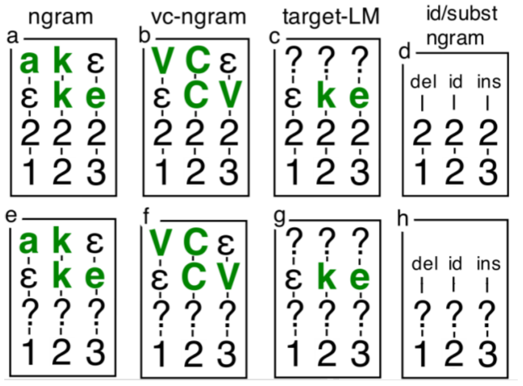


Figure 2: The boxes (a)-(h) represent some of the features that fire on the trigram shown in Fig. 1. These features are explained in detail in section 3.

Here, ℓ_1 indicates that this form pair (*breaking / broke*) as a whole is in a particular cluster, or *word class*, labeled with the arbitrary number $\mathtt{2}$. Notice in Fig. 1 that the class $\mathtt{2}$ is visible in all local windows throughout the string. It allows us to model how certain phenomena, e.g. the vowel change from $\mathtt{ea}$ to $\mathtt{o}$, are more likely in one class than in another. Form pairs in the same class as the *breaking / broke* example might include the following Germanic verbs: *speak*, *break*, *steal*, *tear*, and *bear*.

Of course, word classes are latent (not labeled in our training data). Given x and y, $\boldsymbol{A}_{\boldsymbol{xy}}$ will include alignment strings that specify class $\mathtt{1}$, and others that are identical except that they specify class $\mathtt{2}$; equation (1) sums over both possibilities.[6] In a valid alignment string A, ℓ_1 must be a constant string such as $\mathtt{111}\ldots$ or $\mathtt{222}\ldots$, as in Fig. 1, so that it specifies a single class for the entire form pair. See sections 4.2 and 4.3 for examples of what classes were learned in our experiments.

The latent string ℓ_2 splits the string pair into numbered *regions*. In a valid alignment string, the region numbers must increase throughout ℓ_2, although numbers may be skipped to permit omitted regions. To guide the model to make a useful division into regions, we also require that identity characters such as $(\mathtt{b}, \mathtt{b})$ fall in even regions while change characters such as $(\mathtt{e}, \mathtt{o})$ (substitutions, deletions, or inser-

[5]We set k to a value between 1 and 3, depending on the tasks, always ensuring that no input/output pairs observed in training are excluded. The insertion restriction does slightly enlarge the FSA U_θ: a state must keep track of the number of consecutive ϵ symbols in the immediately preceding x input, and for a few states, this cannot be determined just from the immediately preceding $(n-1)$-gram. Despite this, we found empirically that our approximation is at least as fast as the exact method of Eisner (2002), who sums around cyclic subnetworks to numerical convergence. Furthermore, our approximation does not require us to detect divergence during training.

[6]The latent class is comparable to the latent variable on the tree root symbol *S* in Matsuzaki et al. (2005).

tions) fall in odd regions.[7] Region numbers must not increase within a sequence of consecutive changes or consecutive identities.[8] In Fig. 1, the start of region 1 is triggered by `e:o`, the start of region 2 by the identity `k:k`, region 3 by ϵ`:e`.

Allowing region numbers to be skipped makes it possible to consistently assign similar labels to similar regions across different training examples. Table 2, for example, shows pairs that contain a vowel change in the middle, some of which contain an additional insertion of `ge` in the begining (*verbinden / verbunden*, *reibt / gerieben*). We expect the model to learn to label the `ge` insertion with a 1 and vowel change with a 3, skipping region 1 in the examples where the `ge` insertion is not present (see section 4.2, Analysis).

In the next section we describe features over these enriched alignment strings.

3 Features

One of the simplest ways of scoring a string is an n-gram model. In our log-linear model (1), we include ngram features $f_i(A)$, each of which counts the occurrences in A of a particular n-gram of alignment characters. The log-linear framework lets us include ngram features of different lengths, a form of backoff smoothing (Wu and Khudanpur, 2000).

We use additional backoff features on alignment strings to capture phonological, morphological, and orthographic generalizations. Examples are found in features (b)-(h) in Fig. 2. Feature (b) matches *vowel and consonant* character classes in the input and output dimensions. In the id/subst ngram feature, we have a similar abstraction, where the character classes ins, del, id, and subst are defined over input/output pairs, to match insertions, deletions, identities (matches), and substitutions.

In string transduction tasks, it is helpful to include a language model of the target. While this can be done by mixing the transduction model with a separate language model, it is desirable to include a target language model within the transduction model. We accomplish this by creating target language model features, such as (c) and (g) from Fig. 2, which ignore the input dimension. We also have features which mirror features (a)-(d) but ignore the latent classes and/or regions (e.g. features (e)-(h)).

Notice that our choice of Σ only permits monotonic, 1-to-1 alignments, following Chen (2003). We may nonetheless favor the 2-to-1 alignment (`ea`,`o`) with bigram features such as (`e`,`o`)(`a`,ϵ). A "collapsed" version of a feature will back off from the specific alignment of the characters within a window: thus, (`ea`,`o`) is itself a feature. Currently, we only include collapsed target language model features. These ignore epsilons introduced by deletions in the alignment, so that collapsed `ok` fires in a window that contains `o`ϵ`k`.

4 Experiments

We evaluate our model on two tasks of morphology generation. Predicting morphological forms has been shown to be useful for machine translation and other tasks.[9] Here we describe two sets of experiments: an inflectional morphology task in which models are trained to transduce verbs from one form into another (section 4.2), and a lemmatization task (section 4.3), in which *any* inflected verb is to be reduced to its root form.

4.1 Training and decoding

We train $\boldsymbol{\theta}$ to maximize the regularized[10] conditional log-likelihood[11]

$$\sum_{(x,y^*)\in\mathcal{C}} \log p_{\boldsymbol{\theta}}(y^* \mid x) + ||\boldsymbol{\theta}||^2/2\sigma^2, \qquad (2)$$

where $\mathcal{C}$ is a supervised training corpus. To maximize (2) during training, we apply the gradient-based optimization method L-BFGS (Liu and Nocedal, 1989).[12]

[7]This strict requirement means, perhaps unfortunately, that a single region cannot accommodate the change `ayc:xyz` unless the two `y`'s are not aligned to each other. It could be relaxed, however, to a prior or an initialization or learning bias.

[8]The two boundary characters `#`, numbered 0 and max (max=6 in our experiments), are neither changes nor identities.

[9]E.g., Toutanova et al. (2008) improve MT performance by *selecting* correct morphological forms from a knowledge source. We instead focus on generalizing from observed forms and *generating* new forms (but see *with rootlist* in Table 3).

[10]The variance σ^2 of the L_2 prior is chosen by optimizing on development data. We are also interested in trying an L_1 prior.

[11]Alternatives would include faster error-driven methods (perceptron, MIRA) and slower max-margin Markov networks.

[12]This worked a bit better than stochastic gradient descent.

To decode a test example x, we wish to find $\hat{y} = \operatorname{argmax}_{y \in \Sigma_y^*} p_{\boldsymbol{\theta}}(y \mid x)$. Constructively, $\hat{y}$ is the highest-probability string in the WFSA $T[x]$, where $T = \pi_x^{-1} \circ U_{\boldsymbol{\theta}} \circ \pi_y$ is the trained transducer that maps x nondeterministically to y. Alas, it is NP-hard to find the highest-probability string in a WFSA, even an acyclic one (Casacuberta and Higuera, 2000). The problem is that the probability of each string y is a sum over many paths in $T[x]$ that reflect different alignments of y to x. Although it is straightforward to use a determinization construction (Mohri, 1997)[13] to collapse these down to a single path per y (so that $\hat{y}$ is easily read off the single best path), determinization can increase the WFSA's size exponentially. We approximate by pruning $T[x]$ back to its 1000-best paths before we determinize.[14]

Since the alignments, classes and regions are not observed in $\mathcal{C}$, we do not enjoy the convex objective function of fully-supervised log-linear models. Training equation (2) therefore converges only to some *local* maximum that depends on the starting point in parameter space. To find a good starting point we employ *staged training*, a technique in which several models of ascending complexity are trained consecutively. The parameters of each more complex model are initialized with the trained parameters of the previous simpler model.

Our training is done in four stages. All weights are initialized to zero. ① We first train only features that fire on unigrams of alignment characters, ignoring features that examine the latent strings or backed-off versions of the alignment characters (such as vowel/consonant or target language model features). The resulting model is equivalent to weighted edit distance (Ristad and Yianilos, 1998). ② Next,[15] we train all n-grams of alignment characters, including higher-order n-grams, but no backed-off features or features that refer to latent strings.

13SIA.	lieb**te**,	pick**te**,	red**ete**,	r**ieb**,	tr**ieb**,	zu**zog**
13SKE.	lieb**e**,	pick**e**,	red**e**,	r**eibe**,	tr**eibe**,	zu**ziehe**
2PIE.	lieb**t**,	pick**t**,	red**et**,	reib**t**,	treib**t**,	zuzieh**t**
13PKE.	lieb**en**,	pick**en**,	red**en**,	reib**en**,	treib**en**,	zuzieh**en**
2PKE.	abbreche**t**,	entgegentrete**t**,	zuziehe**t**			
z.	ab**zu**breche**n**,	entgegen**zu**trete**n**,	zu**zu**ziehe**n**			
rP.	redet,	r**eibt**,	tr**eibt**,	verb**i**nde**t**,	überfischt	
pA.	**ge**redet,	**gerieben**,	**ge**tr**ieben**,	verb**u**nde**n**,	überfischt	

Table 2: CELEX forms used in our experiments. Changes from one form to the other are in bold (information not given in training). The changes from rP to pA are very complex. Note also the differing positions of `zu` in z.

③ Next, we add backed-off features as well as all collapsed features. ④ Finally, we train all features. In our experiments, we permitted latent classes `1`–`2` and, where regions are used, regions `0`–`6`. For speed, stages ②–④ used a pruned Σ that included only "plausible" alignment characters: a may not align to b unless it did so in the trained stage-(1) model's optimal alignment of *at least one* training pair (x, y^*).

4.2 Inflectional morphology

We conducted several experiments on the CELEX morphological database. We arbitrarily considered mapping the following German verb forms:[16] 13SIA $\rightarrow$ 13SKE, 2PIE $\rightarrow$ 13PKE, 2PKE $\rightarrow$ z, and rP $\rightarrow$ pA.[17] We refer to these tasks as 13SIA, 2PIE, 2PKE and rP. Table 2 shows some examples of regular and irregular forms. Common phenomena include stem changes (`ei:ie`), prefixes inserted after other morphemes (*ab**zu**brechen*) and circumfixes (***ge**rieb**en***).

We compile lists of form pairs from CELEX. For each task, we sample 2500 data pairs without replacement, of which 500 are used for training, 1000 as development and the remaining 1000 as test data. We train and evaluate models on this data and repeat

[13]Weighted determinization is not always possible, but it is in our case because our limit to k consecutive insertions guarantees that $T[x]$ is acyclic.

[14]This value is high enough; we see no degradations in performance if we use only 100 or even 10 best paths. Below that, performance starts to drop slightly. In both of our tasks, our conditional distributions are usually peaked: the 5 best output candidates amass $> 99\%$ of the probability mass on average. Entropy is reduced by latent classes and/or regions.

[15]When unclamping a feature at the start of stages ②–④, we initialize it to a random value from $[-0.01, 0.01]$.

[16]From the available languages in CELEX (German, Dutch, and English), we selected German as the language with the most interesting morphological phenomena, leaving the multilingual comparison for the lemmatization task (section 4.3), where there were previous results to compare with. The 4 German datasets were picked arbitrarily.

[17]A key to these names: 13SIA=*1st/3rd sg. ind. past*; 13SKE=*1st/3rd sg. subjunct. pres.*; 2PIE=*2nd pl. ind. pres.*; 13PKE=*1st/3rd pl. subjunct. pres.*; 2PKE=*2nd pl. subjunct. pres.*; z=*infinitive*; rP=*imperative pl.*; pA=*past part.*

	Features							Task			
	ng	vc	tlm	tlm-coll	id	lat.cl.	lat.reg.	13SIA	2PIE	2PKE	rP
ngrams	x							82.3 (.23)	88.6 (.11)	74.1 (.52)	70.1 (.66)
ngrams+x	x				x			82.8 (.21)	88.9 (.11)	74.3 (.52)	70.0 (.68)
	x		x					82.0 (.23)	88.7 (.11)	74.8 (.50)	69.8 (.67)
	x		x		x			82.5 (.22)	88.6 (.11)	74.9 (.50)	70.0 (.67)
	x		x	x				81.2 (.24)	88.7 (.11)	74.5 (.50)	68.6 (.69)
	x		x	x	x			82.5 (.22)	88.8 (.11)	74.5 (.50)	69.2 (.69)
	x	x						82.4 (.22)	88.9 (.11)	74.8 (.51)	69.9 (.68)
	x	x			x			83.0 (.21)	88.9 (.11)	74.9 (.50)	70.3 (.67)
	x	x	x					82.2 (.22)	88.8 (.11)	74.8 (.50)	70.0 (.67)
	x	x	x		x			82.9 (.21)	88.6 (.11)	75.2 (.50)	69.7 (.68)
	x	x	x	x				81.9 (.23)	88.6 (.11)	74.4 (.51)	69.1 (.68)
	x	x	x	x	x			82.8 (.21)	88.7 (.11)	74.7 (.50)	69.9 (.67)
ngrams+x +latent	x	x	x	x	x	x		84.8 (.19)	**93.6 (.06)**	75.7 (.48)	81.8 (.43)
	x	x	x	x	x		x	**87.4 (.16)**	**93.8 (.06)**	**88.0 (.28)**	83.7 (.42)
	x	x	x	x	x	x	x	**87.5 (.16)**	93.4 (.07)	**87.4 (.28)**	**84.9 (.39)**
Moses3								73.9 (.40)	92.0 (.09)	67.1 (.70)	67.6 (.77)
Moses9								85.0 (.21)	**94.0 (.06)**	82.3 (.31)	70.8 (.67)
Moses15								85.3 (.21)	**94.0 (.06)**	82.8 (.30)	70.8 (.67)

Table 1: Exact-match accuracy and average edit distance (the latter in parentheses) versus the correct answer on the German inflection task, using different combinations of feature classes. The label *ngrams* corresponds to the second stage of training, *ngrams+x* to the third where backoff features may fire (vc = vowel/consonant, tlm = target LM, tlm-coll = collapsed tlm, id = identity/substitution/deletion features), and *ngrams+x+latent* to the fourth where features sensitive to latent classes and latent regions are allowed to fire. The highest n-gram order used is 3, except for *Moses9* and *Moses15* which examine windows of up to 9 and 15 characters, respectively. We mark in **bold** the best result for each dataset, along with all results that are statistically indistinguishable (paired permutation test, $p < 0.05$).

the process 5 times. All results are averaged over these 5 runs.

Table 1 and Fig. 3 report separate results after stages ②, ③, and ④ of training, which include successively larger feature sets. These are respectively labeled *ngrams*, *ngrams+x*, and *ngrams+x+latent*. In Table 1, the last row in each section shows the full feature set at that stage (cf. Fig. 3), while earlier rows test feature subsets.[18]

Our baseline is the SMT toolkit Moses (Koehn et al., 2007) run over letter strings rather than word strings. It is trained (on the same data splits) to find substring-to-substring phrase pairs and translate from one form into another (with phrase reordering turned off). Results reported as *moses3* are obtained from Moses runs that are constrained to the same context windows that our models use, so the maximum phrase length and the order of the target language model were set to 3. We also report results using much larger windows, *moses9* and *moses15*.

[18]The number k of consecutive insertions was set to 3.

Results. The results in Table 1 show that including *latent classes and/or regions* improves the results dramatically. Compare the last line in *ngrams+x* to the last line in *ngrams+x+latent*. The accuracy numbers improve from 82.8 to 87.5 (13SIA), from 88.7 to 93.4 (2PIE), from 74.7 to 87.4 (2PKE), and from 69.9 to 84.9 (rP).[19] This shows that error reductions between 27% and 50% were reached. On 3 of 4 tasks, even our simplest *ngrams* method beats the *moses3* method that looks at the same amount of context.[20] With our full model, in particular using latent features, we always outperform *moses3*—and even outperform *moses15* on 3 of the 4 datasets, reducing the error rate by up to 48.3% (rP). On the fourth task (2PIE), our method and *moses15* are statistically tied. *Moses15* has access to context windows of five times the size than we allowed our methods in our experiments.

[19]All claims in the text are statistically significant under a paired permutation test ($p < .05$).

[20]This bears out our contention in footnote 2 that a "segmenting" channel model is damaging. Moses cannot fully recover by using overlapping windows in the *language* model.

While the gains from *backoff features* in Table 1 were modest (significant gains only on 13SIA), the learning curve in Fig. 3 suggests that they were helpful for smaller training sets on 2PKE (see *ngrams* vs *ngrams+x* on 50 and 100) and helped consistently over different amounts of training data for 13SIA.

Analysis. The types of errors that our system (and the moses baseline) make differ from task to task. Due to lack of space, we mainly focus on the complex rP task. Here, most errors come from wrongly copying the input to the output, without making a change (40-50% of the errors in all models, except for our model with latent classes and no regions, where it accounts for only 30% of the errors). This is so common because about half of the training examples contain identical inputs and outputs (as in the imperative *berechnet* and the participle *(ihr habt) berechnet*). Another common error is to wrongly assume a regular conjugation (just insert the prefix *ge-* at the beginning). Interestingly, this error by simplification is more common in the Moses models (44% of moses3 errors, down to 40% for moses15) than in our models, where it accounts for 37% of the errors of our *ngrams* model and only 19% if latent classes *or* latent regions are used; however, it goes up to 27% if both latent classes and regions are used.[21] All models for rP contain errors where wrong analogies to observed words are made (*verschweisst/verschwissen* in analogy to the observed *durchweicht/durchwichen*, or *bebt/geboben* in analogy to *hebt/gehoben*). In the 2PKE task, most errors result from inserting the *zu* morpheme at a wrong place or inserting two of them, which is always wrong. This error type was greatly reduced by latent regions, which can discover different parameters for different positions, making it easier to identify where to insert the *zu*.

Analysis of the 2 latent classes (when used) shows that a *split into regular and irregular conjugations* has been learned. For the rP task we compute, for each data pair in development data, the posterior probabilities of membership in one or the other class. 98% of the regular forms, in which the past participle is built with *ge-* ... *-t*, fall into one class,

[21]We suspect that training of the models that use classes and regions together was hurt by the increased non-convexity; annealing or better initialization might help.

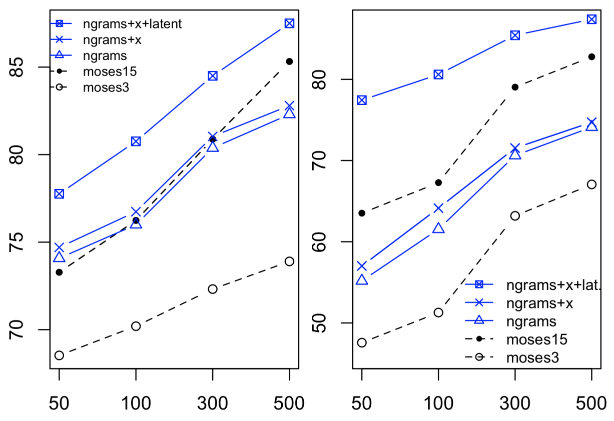


Figure 3: Learning curves for German inflection tasks, 13SIA (left) and 2PKE (right), as a function of the number of training pairs. *ngrams+x* means all backoff features were used, *ngrams+x+latent* means all latent features were used in addition. *Moses15* examines windows of up to 15 characters.

which in turn consists nearly exclusively (96%) of these forms. Different irregular forms are lumped into the other class.

The learned regions are consistent across different pairs. On development data for the rP task, 94.3% of all regions that are labeled 1 are the insertion sequence (ϵ,ge), region 3 consists of vowel changes 93.7% of the time; region 5 represents the typical suffixes (t, en), (et, en), (t, n) (92.7%). In the *2PKE* task, region 0 contains different prefixes (e.g. *entgegen* in *entgegenzutreten*), regions 1 and 2 are empty, region 3 contains the *zu* affix, region 4 the stem, and region 5 contains the suffix.

The pruned alignment alphabet excluded a few gold standard outputs so that the model contains paths for 98.9%–99.9% of the test examples. We verified that the insertion limit did not hurt oracle accuracy.

4.3 Lemmatization

We apply our models to the task of lemmatization, where the goal is to generate the lemma given an inflected word form. We compare our model to Wicentowski (2002, chapter 3), an alternative supervised approach. Wicentowski's *Base* model simply learns how to replace an arbitrarily long suffix string of an input word, choosing some previously observed suffix $\rightarrow$ suffix replacement based on the input word's

	Without rootlist (generation)						With rootlist (selection)					
	Wicentowski (2002)			This paper			Wicentowski (2002)			This paper		
Lang.	Base	Af.	WFA.	n	n+x	n+x+l	Base	Af.	WFA.	n	n+x	n+x+l
Basque	85.3	81.2	80.1	91.0 (.20)	91.1 (.20)	**93.6 (.14)**	94.5	94.0	**95.0**	90.9 (.29)	90.8 (.31)	90.9 (.30)
English	91.0	94.7	93.1	92.4 (.09)	93.4 (.08)	**96.9 (.05)**	98.3	98.6	98.6	**98.7 (.04)**	**98.7(.04)**	**98.7(.04)**
Irish	43.3	-	70.8	96.8 (.07)	97.0 (.06)	**97.8 (.04)**	43.9	-	89.1	**99.6 (.02)**	**99.6 (.02)**	**99.5 (.03)**
Tagalog	0.3	80.3	81.7	80.5 (.32)	83.0 (.29)	**88.6 (.19)**	0.8	91.8	96.0	97.0 (.07)	97.2 (.07)	**97.7 (.05)**

Table 3: Exact-match accuracy and average edit distance (the latter in parentheses) on the 8 lemmatization tasks (2 tasks $\times$ 4 languages). The numbers from Wicentowski (2002) are for his Base, Affix and WFAffix models. The numbers for our models are for the feature sets *ngrams*, *ngrams+x*, *ngrams+x+latent*. The best result per task is in **bold** (as are statistically indistinguishable results when we can do the comparison, i.e., for our own models). Corpus sizes: Basque 5,842, English 4,915, Irish 1,376, Tagalog 9,479.

final n characters (interpolating across different values of n). His *Affix* model essentially applies the Base model after stripping canonical prefixes and suffixes (given by a user-supplied list) from the input and output. Finally, his *WFAffix* uses similar methods to also learn substring replacements for a stem vowel cluster and other linguistically significant regions in the form (identified by a *deterministic* alignment and segmentation of training pairs). This approach is a bit like our change regions combined with Moses's region-independent phrase pairs.

We compare against all three models. Note that *Affix* and *WFAffix* have an advantage that our models do not, namely, user-supplied lists of canonical affixes for each language. It is interesting to see how our models with their more non-committal trigram structure compare to this. Table 3 reports results on the data sets used in Wicentowski (2002), for Basque, English, Irish, and Tagalog. Following Wicentowski, 10-fold cross-validation was used. The columns *n+x* and *n+x+l* mean *ngram+x* and *ngram+x+latent*, respectively. As latent variables, we include 2 word classes but no change regions.[22]

For completeness, Table 3 also compares with Wicentowski (2002) on a *selection* (rather than generation) task. Here, at test time, the lemma is selected from a candidate list of known lemmas, namely, all the output forms that appeared in training data.[23] These additional results are labeled *with rootlist* in the right half of Table 3.

On the supervised generation task *without rootlist*, our models outperform Wicentowski (2002) by a large margin. Comparing our results that use latent classes (*n+x+l*) with Wicentowski's best models we observe error reductions ranging from about 38% (Tagalog) to 92% (Irish). On the selection task *with rootlist*, we outperform Wicentowski (2002) in English, Irish, and Tagalog.

Analysis. We examined the classes learned on English lemmatization by our *ngrams+x+latent* model. For each of the input/output pairs in development data, we found the most probable latent class. For the most part, the 2 classes are separated based on whether or not the correct output ends in e. This use of latent classes helped address many errors like *wronging / wronge* or *owed / ow*). Such missing or surplus final e's account for 72.5% of the errors for *ngrams* and 70.6% of the errors for *ngrams+x*, but only 34.0% of the errors for *ngrams+x+latent*.

The test oracles are between 99.8% – 99.9%, due to the pruned alignment alphabet. As on the inflection task, the insertion limit does not exclude any gold standard paths.

5 Finite-State Feature Implementation

We used the OpenFST library (Allauzen et al., 2007) to implement all finite-state computations, using the expectation semiring (Eisner, 2002) for training.

Our model is defined by the WFSA $U_{\boldsymbol{\theta}}$, which is used to score alignment strings in Σ^* (section 2.2). We now sketch how to construct $U_{\boldsymbol{\theta}}$ from features.

n-gram construction The construction that we currently use is quite simple. All of our current features fire on windows of width $\leq$ 3. We build a WFSA with the structure of a 3-gram language

[22]The insertion limit k was set to 2 for Basque and 1 for the other languages.

[23]Though test data contained no (input, output) pairs from training data, it reused many of the output forms, since many inflected inputs are to be mapped to the same output lemma.

model over Σ^*. Each of the $|\Sigma|^2$ states remembers two previous alignment characters ab of history; for each $c \in \Sigma$, it has an outgoing arc that accepts c (and leads to state bc). The weight of this arc is the total weight (from θ) of the small set of features that fire when the trigram window includes abc. By convention, these also include features on bc and c (which may be regarded as backoff features $?bc$ and $??c$). Since each character in Σ is actually a 4-tuple, this trigram machine is fairly large. We build it lazily ("on the fly"), constructing arcs only as needed to deal with training or test data.

Feature templates Our experiments use over 50,000 features. How do we *specify* these features to the above construction? Rather than writing ordinary code to extract features from a window, we find it convenient to harness FSTs as a "little language" (Bentley, 1986) for specifying entire sets of features.

A **feature template** T is an nondeterministic FST that maps the contents of the sliding window, such as `abc`, to one or more **features**, which are also described as strings.[24] The n-gram machine described above can compute $T[((\mathtt{a}^?\mathtt{b})^?\mathtt{c})^?]$ to find out what features fire on `abc` and its suffixes. One simple feature template performs "vowel/consonant backoff"; e.g., it maps `abc` to the feature named `VCC`. Fig. 2 showed the result of applying several actual feature templates to the window shown in Fig. 1. The extended regular expression calculus provides a flexible and concise notation for writing down these FSTs. As a trivial example, the trigram "vowel/consonant backoff" transducer can be described as $T = VVV$, where V is a transducer that performs backoff on a *single* alignment character. Feature templates should make it easy to experiment with adding various kinds of linguistic knowledge. We have additional algorithms for compiling $U_{\boldsymbol{\theta}}$ from a set of *arbitrary* feature templates,[25] including templates whose features consider windows of variable or even unbounded width. The details are beyond the scope of this paper, but it is worth pointing out that they exploit the fact that feature templates are FSTs and not arbitrary code.

[24]Formally, if i is a string naming a feature, then $f_i(A)$ counts the number of positions in A that are immediately preceded by some string in $T^{-1}[i]$.

[25]Provided that the total number of features is finite.

6 Conclusions

The modeling framework we have presented here is, we believe, an attractive solution to most string transduction problems in NLP. Rather than learn the topology of an arbitrary WFST, one specifies the topology using a small set of feature templates, and simply trains the weights.

We evaluated on two morphology generation tasks. When inflecting German verbs we, even with the simplest features, outperform the *moses3* baseline on 3 out of 4 tasks, which uses the same amount of context as our models. Introducing more sophisticated features that have access to latent classes and regions improves our results dramatically, even on small training data sizes. Using these we outperform *moses9* and *moses15*, which use long context windows, reducing error rates by up to 48%. On the lemmatization task we were able to improve the results reported in Wicentowski (2002) on three out of four tested languages and reduce the error rates by 38% to 92%. The model's errors are often reasonable misgeneralizations (e.g., assume regular conjugation where irregular would have been correct), and it is able to use even a small number of latent variables (including the latent alignment) to capture useful linguistic properties.

In future work, we would like to identify a set of features, latent variables, and training methods that port well across languages and string-transduction tasks. We would like to use features that look at wide context on the *input* side, which is inexpensive (Jiampojamarn et al., 2007). Latent variables we wish to consider are an increased number of word classes; more flexible regions—see Petrov et al. (2007) on learning a state transition diagram for *acoustic* regions in phone recognition—and phonological features and syllable boundaries. Indeed, our local log-linear features over several aligned latent strings closely resemble the soft constraints used by phonologists (Eisner, 1997). Finally, rather than define a fixed set of feature templates as in Fig. 2, we would like to refine empirically useful features during training, resulting in language-specific backoff patterns and adaptively sized n-gram windows. Many of these enhancements will increase the computational burden, and we are interested in strategies to mitigate this, including approximation methods.

References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *Proc. of CIAA*, volume 4783.

Jon Bentley. 1986. Programming pearls [column]. *Communications of the ACM*, 29(8), August.

Maximilian Bisani and Hermann Ney. 2002. Investigations on jointmultigram models for grapheme-to-phoneme conversion.

Francisco Casacuberta and Colin De La Higuera. 2000. Computational complexity of problems on probabilistic grammars and transducers. In *Proc. of the 5th International Colloquium on Grammatical Inference: Algorithms and Applications*, volume 1891.

Francisco Casacuberta. 2000. Inference of finite-state transducers by using regular grammars and morphisms. In A.L. Oliveira, editor, *Grammatical Inference: Algorithms and Applications*, volume 1891.

Stanley F. Chen. 2003. Conditional and joint models for grapheme-to-phoneme conversion. In *Proc. of Interspeech*.

Alexander Clark. 2001. Learning morphology with Pair Hidden Markov Models. In *Proc. of the Student Workshop at the 39th Annual Meeting of the Association for Computational Linguistics*, Toulouse, France, July.

Sabine Deligne, Francois Yvon, and Frédéric Bimbot. 1995. Variable-length sequence matching for phonetic transcription using joint multigrams. In *Eurospeech*.

Vera Demberg, Helmut Schmid, and Gregor Möhler. 2007. Phonological constraints and morphological preprocessing for grapheme-to-phoneme conversion. In *Proc. of ACL*, Prague, Czech Republic, June.

Jason Eisner. 1997. Efficient generation in primitive Optimality Theory. In *Proc. of ACL-EACL*, Madrid, July.

Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proc. of ACL*.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *Proceedings of ACL-08: HLT*, pages 959–967, Columbus, Ohio, June. Association for Computational Linguistics.

Dayne Freitag and Shahram Khadivi. 2007. A sequence alignment model based on the averaged perceptron. In *Proc. of EMNLP-CoNLL*.

Lucian Galescu and James F. Allen. 2001. Bi-directional conversion between graphemes and phonemes using a joint N-gram model.

Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. In *Proc. of NAACL-HLT*, Rochester, New York, April.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL, Companion Volume*, Prague, Czech Republic, June.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Math. Programming*, 45(3, (Ser. B)):503–528.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proc. of ACL*, Ann Arbor, Michigan, June.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2).

Slav Petrov, Adam Pauls, and Dan Klein. 2007. Learning structured models for phone recognition. In *Proc. of EMNLP-CoNLL*.

Eric Sven Ristad and Peter N. Yianilos. 1998. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5).

Holger Schwenk, Marta R. Costa-jussa, and Jose A. R. Fonollosa. 2007. Smooth bilingual n-gram translation. In *Proc. of EMNLP-CoNLL*, pages 430–438.

Tarek Sherif and Grzegorz Kondrak. 2007. Substring-based transliteration. In *Proc. of ACL*, Prague, Czech Republic, June.

Kristina Toutanova, Hisami Suzuki, and Achim Ruopp. 2008. Applying morphology generation models to machine translation. In *Proceedings of ACL-08: HLT*, Columbus, Ohio, June.

Richard Wicentowski. 2002. *Modeling and Learning Multilingual Inflectional Morphology in a Minimally Supervised Framework*. Ph.D. thesis, Johns-Hopkins University.

Jun Wu and Sanjeev Khudanpur. 2000. Efficient training methods for maximum entropy language modeling. In *Proc. of ICSLP*, volume 3, Beijing, October.

Bing Zhao, Nguyen Bach, Ian Lane, and Stephan Vogel. 2007. A log-linear block transliteration model based on bi-stream HMMs. In *Proc. of NAACL-HLT*, Rochester, New York, April.

Soft-Supervised Learning for Text Classification

Amarnag Subramanya & Jeff Bilmes
Dept. of Electrical Engineering,
University of Washington, Seattle, WA 98195, USA.
{asubram,bilmes}@ee.washington.edu

Abstract

We propose a new graph-based semi-supervised learning (SSL) algorithm and demonstrate its application to document categorization. Each document is represented by a vertex within a weighted undirected graph and our proposed framework minimizes the weighted Kullback-Leibler divergence between distributions that encode the class membership probabilities of each vertex. The proposed objective is convex with guaranteed convergence using an alternating minimization procedure. Further, it generalizes in a straightforward manner to multi-class problems. We present results on two standard tasks, namely Reuters-21578 and WebKB, showing that the proposed algorithm significantly outperforms the state-of-the-art.

1 Introduction

Semi-supervised learning (SSL) employs small amounts of labeled data with relatively large amounts of unlabeled data to train classifiers. In many problems, such as speech recognition, document classification, and sentiment recognition, annotating training data is both time-consuming and tedious, while unlabeled data are easily obtained thus making these problems useful applications of SSL. Classic examples of SSL algorithms include self-training (Yarowsky, 1995) and co-training (Blum and Mitchell, 1998). Graph-based SSL algorithms are an important class of SSL techniques that have attracted much of attention of late (Blum and Chawla, 2001; Zhu et al., 2003). Here one assumes that the data (both labeled and unlabeled) is embedded within a low-dimensional manifold expressed by a graph. In other words, each data sample is represented by a vertex within a weighted graph with the weights providing a measure of similarity between vertices.

Most graph-based SSL algorithms fall under one of two categories – those that use the graph structure to spread labels from labeled to unlabeled samples (Szummer and Jaakkola, 2001; Zhu and Ghahramani, 2002) and those that optimize a loss function based on smoothness constraints derived from the graph (Blum and Chawla, 2001; Zhu et al., 2003; Joachims, 2003; Belkin et al., 2005). Sometimes the two categories are similar in that they can be shown to optimize the same underlying objective (Zhu and Ghahramani, 2002; Zhu et al., 2003). In general graph-based SSL algorithms are non-parametric and transductive.[1] A learning algorithm is said to be transductive if it is expected to work only on a closed data set, where a test set is revealed at the time of training. In practice, however, transductive learners can be modified to handle unseen data (Zhu, 2005a; Sindhwani et al., 2005). A common drawback of many graph-based SSL algorithms (e.g. (Blum and Chawla, 2001; Joachims, 2003; Belkin et al., 2005)) is that they assume binary classification tasks and thus require the use of sub-optimal (and often computationally expensive) approaches such as one vs. rest to solve multi-class problems, let alone structured domains such as strings and trees. There are also issues related to degenerate solutions (all unlabeled samples classified as belonging to a single

[1]Excluding Manifold Regularization (Belkin et al., 2005).

class) (Blum and Chawla, 2001; Joachims, 2003; Zhu and Ghahramani, 2002). For more background on graph-based and general SSL and their applications, see (Zhu, 2005a; Chapelle et al., 2007; Blitzer and Zhu, 2008).

In this paper we propose a new algorithm for graph-based SSL and use the task of text classification to demonstrate its benefits over the current state-of-the-art. Text classification involves automatically assigning a given document to a fixed number of semantic categories. Each document may belong to one, many, or none of the categories. In general, text classification is a *multi-class* problem (more than 2 categories). Training fully-supervised text classifiers requires large amounts of labeled data whose annotation can be expensive (Dumais et al., 1998). As a result there has been interest is using SSL techniques for text classification (Joachims, 1999; Joachims, 2003). However past work in semi-supervised text classification has relied primarily on one vs. rest approaches to overcome the inherent multi-class nature of this problem. We believe such an approach may be sub-optimal because, disregarding data overlap, the different classifiers have training procedures that are independent of one other. In order to address the above drawback we propose a new framework based on optimizing a loss function composed of Kullback-Leibler divergence (KL-divergence) (Cover and Thomas, 1991) terms between probability distributions defined for each graph vertex. The use of probability distributions, rather than fixed integer labels, not only leads to a straightforward multi-class generalization, but also allows us to exploit other well-defined functions of distributions, such as entropy, to improve system performance and to allow for the measure of uncertainty. For example, with a single integer, at most all we know is its assignment. With a distribution, we can continuously move from knowing an assignment with certainty (i.e., an entropy of zero) to expressions of doubt or multiple valid possibilities (i.e., an entropy greater than zero). This is particularly useful for document classification as we will see. We also show how one can use the alternating minimization (Csiszar and Tusnady, 1984) algorithm to optimize our objective leading to a relatively simple, fast, easy-to-implement, guaranteed to converge, iterative, and closed form update for each iteration.

2 Proposed Graph-Based Learning Framework

We consider the transductive learning problem, i.e., given a training set $\mathcal{D} = \{\mathcal{D}_l, \mathcal{D}_u\}$, where $\mathcal{D}_l$ and $\mathcal{D}_u$ are the sets of labeled and unlabeled samples respectively, the task is to infer the labels for the samples in $\mathcal{D}_u$. In other words, $\mathcal{D}_u$ is the "test-set." Here $\mathcal{D}_l = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, $\mathcal{D}_u = \{\mathbf{x}_i\}_{i=l+1}^{l+u}$, $\mathbf{x}_i \in \mathrm{X}$ (the input space of the classifier, and corresponds to vectors of features) and $y_i \in \mathrm{Y}$ (the space of classifier outputs, and for our case is the space of non-negative integers). Thus $|\mathrm{Y}| = 2$ yields binary classification while $|\mathrm{Y}| > 2$ yields multi-class. We define $n = l + u$, the total number of samples in the training set. Given $\mathcal{D}$, most graph-based SSL algorithms utilize an undirected weighted graph $\mathcal{G} = (V, E)$ where $V = \{1, \ldots, n\}$ are the data points in $\mathcal{D}$ and $E = V \times V$ are the set of undirected edges between vertices. We use $w_{ij} \in \mathbf{W}$ to denote the weight of the edge between vertices i and j. $\mathbf{W}$ is referred to as the weight (or affinity) matrix of $\mathcal{G}$. As will be seen shortly, the input features $\mathbf{x}_i$ effect the final classification results via $\mathbf{W}$, i.e., the graph. Thus graph construction is crucial to the success of any graph-based SSL algorithm. Graph construction "is more of an art, than science" (Zhu, 2005b) and is an active research area (Alexandrescu and Kirchhoff, 2007). In general the weights are formed as $w_{ij} = \mathrm{sim}(\mathbf{x}_i, \mathbf{x}_j)\delta(j \in \mathcal{K}(i))$. Here $\mathcal{K}(i)$ is the set of i's k-nearest-neighbors ($\mathcal{K}$NN), $\mathrm{sim}(\mathbf{x}_i, \mathbf{x}_j)$ is a given measure of similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$, and $\delta(c)$ returns a 1 if c is true and 0 otherwise. Getting the similarity measure right is crucial for the success of any SSL algorithm as that is what determines the graph. Note that setting $\mathcal{K}(i) = |V| = n$ results in a fully-connected graph. Some popular similarity measures include

$$\mathrm{sim}(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{\sigma^2}} \text{ or}$$

$$\mathrm{sim}(\mathbf{x}_i, \mathbf{x}_j) = \cos(\mathbf{x}_i, \mathbf{x}_j) = \frac{\langle \mathbf{x}_i, \mathbf{x}_j \rangle}{\| \mathbf{x}_i \|_2^2 \| \mathbf{x}_j \|_2^2}$$

where $\| \mathbf{x}_i \|_2$ is the $\mathcal{L}_2$ norm, and $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ is the inner product of $\mathbf{x}_i$ and $\mathbf{x}_j$. The first similarity measure is an RBF kernel applied on the squared Euclidean distance while the second is cosine similarity. In this paper *all* graphs are constructed using cosine similarity.

We next introduce our proposed approach. For every $i \in V$, we define a probability distribution p_i over the elements of Y. In addition let $r_j, j = 1 \ldots l$ be another set of probability distributions again over the elements of Y (recall, Y is the space of classifier outputs). Here $\{r_j\}_j$ represents the labels of the supervised portion of the training data. If the label for a given labeled data point consists only of a single integer, then the entropy of the corresponding r_j is zero (the probability of that integer will be unity, with the remaining probabilities being zero). If, on the other hand, the "label" for a given labeled data point consists of a set of integers (e.g., if the object is a member of multiple classes), then r_j is able to represent this property accordingly (see below). We emphasize again that both p_i and r_j are probability distributions, with r_j fixed throughout training. The goal of learning in this paper is to find the best set of distributions $p_i, \forall i$ that attempt to: 1) agree with the labeled data r_j wherever it is available; 2) agree with each other (when they are close according to a graph); and 3) be smooth in some way. These criteria are captured in the following *new* multi-class SSL optimization procedure:

$$\min_{\mathrm{p}} C_1(\mathrm{p}), \text{ where } C_1(\mathrm{p}) = \left[\sum_{i=1}^{l} D_{KL}\big(r_i || p_i\big) + \mu \sum_{i}^{n} \sum_{j} w_{ij} D_{KL}\big(p_i || p_j\big) - \nu \sum_{i=1}^{n} H(p_i) \right], \tag{1}$$

and where $\mathrm{p} \triangleq (p_1, \ldots, p_n)$ denotes the entire set of distributions to be learned, $H(p_i) = -\sum_{\mathrm{y}} p_i(\mathrm{y}) \log p_i(\mathrm{y})$ is the standard Shannon entropy function of p_i, $D_{KL}(p_i || q_j)$ is the KL-divergence between p_i and q_j, and μ and ν are hyperparameters whose selection we discuss in section 5. The distributions r_i are derived from $\mathcal{D}_l$ (as mentioned above) and this can be done in one of the following ways: (a) if $\hat{y}_i$ is the single supervised label for input $\mathbf{x}_i$ then $r_i(y) = \delta(y = \hat{y}_i)$, which means that r_i gives unity probability for y equaling the label $\hat{y}_i$; (b) if $\hat{y}_i = \{\hat{y}_i^{(1)}, \ldots, \hat{y}_i^{(k)}\}$, $k \leq |\mathrm{Y}|$ is a *set* of possible outputs for input $\mathbf{x}_i$, meaning an object validly falls into all of the corresponding categories, we set $r_i(y) = (1/k)\delta(y \in \hat{y}_i)$ meaning that r_i is uniform over only the possible categories and zero otherwise; (c) if the labels are somehow provided in the form of a set of non-negative scores, or even a probability distribution itself, we just set r_i to be equal to those scores (possibly) normalized to become a valid probability distribution. Among these three cases, case (b) is particularly relevant to text classification as a given document many belong to (and in practice may be labeled as) many classes. The final classification results, i.e., the final labels for $\mathcal{D}_u$, are then given by $\hat{y} = \operatorname*{argmax}_{\mathrm{y} \in \mathrm{Y}} p_i(\mathrm{y})$.

We next provide further intuition on our objective function. SSL on a graph consists of finding a labeling $\mathcal{D}_u$ that is consistent with both the labels provided in $\mathcal{D}_l$ and the geometry of the data induced by the graph. The first term of C_1 will penalize the solution p_i $i \in \{1, \ldots, l\}$, when it is far away from the labeled training data $\mathcal{D}_l$, but it does not insist that $p_i = r_i$, as allowing for deviations from r_i can help especially with noisy labels (Bengio et al., 2007) or when the graph is extremely dense in certain regions. As explained above, our framework allows for the case where supervised training is uncertain or ambiguous. We consider it reasonable to call our approach *soft-supervised* learning, generalizing the notion of semi-supervised learning, since there is even more of a continuum here between fully supervised and fully unsupervised learning than what typically exists with SSL. Soft-supervised learning allows uncertainty to be expressed (via a probability distribution) about any of the labels individually.

The second term of C_1 penalizes a lack of consistency with the geometry of the data and can be seen as a graph regularizer. If w_{ij} is large, we prefer a solution in which p_i and p_j are close in the KL-divergence sense. While KL-divergence is asymmetric, given that $\mathcal{G}$ is undirected implies $\mathbf{W}$ is symmetric ($w_{ij} = w_{ji}$) and as a result the second term is inherently symmetric.

The last term encourages each p_i to be close to the uniform distribution if not preferred to the contrary by the first two terms. This acts as a guard against degenerate solutions commonly encountered in SSL (Blum and Chawla, 2001; Joachims, 2003). For example, consider the case where part of the graph is almost completely disconnected from any labeled vertex (which is possible in the k-nearest neighbor case). In such situations the third term ensures that the nodes in this disconnected region are encouraged to yield a uniform distribution, validly expressing the fact that we do not know the labels of these nodes based on the nature of the graph. More generally, we conjecture that by maximizing the entropy of each p_i, the classifier has a better chance of producing high entropy results in graph regions of low confidence (e.g. close to the decision boundary and/or low density regions). This overcomes a common drawback of a large number of state-of-the-art classifiers that tend to be confident even in regions close to the decision boundary.

We conclude this section by summarizing some of the features of our proposed framework. It should be clear that C_1 uses the “manifold assumption” for SSL (see chapter 2 in (Chapelle et al., 2007)) — it assumes that the input data can be embedded within a low-dimensional manifold (the graph). As the objective is defined in terms of probability distributions over integers rather than just integers (or to real-valued relaxations of integers (Joachims, 2003; Zhu et al., 2003)), the framework generalizes in a straightforward manner to multi-class problems. Further, all the parameters are estimated jointly (compare to one vs. rest approaches which involve solving $|\mathrm{Y}|$ independent problems). Furthermore, the objective is capable of handling label training data uncertainty (Pearl, 1990). Of course, this objective would be useless if it wasn't possible to efficiently and easily optimize it on large data sets. We next describe a method that can do this.

3 Learning with Alternating Minimization

As long as $\mu, \nu \geq 0$, the objective $C_1(\mathrm{p})$ is convex. This follows since $D_{KL}(p_i||p_j)$ is convex in the pair (p_i, p_j) (Cover and Thomas, 1991), negative entropy is convex, and a positive-weighted linear combination of a set of convex functions is convex. Thus, the problem of minimizing C_1 over the space of collections of probability distributions (a convex set) constitutes a convex programming problem (Bertsekas, 2004). This property is *extremely* beneficial since there is a unique global optimum and there are a variety of methods that can be used to yield that global optimum. One possible method might take the derivative of the objective along with Lagrange multipliers to ensure that we stay within the space of probability distributions. This method can sometimes yield a closed form single-step analytical expression for the globally optimum solution. Unfortunately, however, our problem does not admit such a closed form solution because the gradient of $C_1(\mathrm{p})$ with respect to $p_i(\mathrm{y})$ is of the form, $k_1 p_i(\mathrm{y}) \log p_i(\mathrm{y}) + k_2 p_i(\mathrm{y}) + k_3$ (where $k_1,\ k_2,\ k_3$ are fixed constants). Sometimes, optimizing the dual of the objective can also produce a solution, but unfortunately again the dual of our objective also does not yield a closed form solution. The typical next step, then, is to resort to iterative techniques such as gradient descent along with modifications to ensure that the solution stays within the set of probability distributions (the gradient of C_1 alone will not necessarily point in the direction where p is still a valid distribution) - one such modification is called the method of multipliers (MOM). Another solution would be to use computationally complex (and complicated) algorithms like interior point methods (IPM). While all of the above methods (described in detail in (Bertsekas, 2004)) are feasible ways to solve our problem, they each have their own drawbacks. Using MOM, for example, requires the careful tuning of a number of additional parameters such as learning rates, growth factors, and so on. IPM involves inverting a matrix of the order of the number of variables and constraints during each iteration.

We instead adopt a different strategy based on alternating minimization (Csiszar and Tusnady, 1984). This approach has a single additional optimization parameter (contrasted with MOM), admits a closed form solution for each iteration not involving any matrix inversion (contrasted with IPM), and yields guaranteed convergence to the global optimum. In order to render our approach amenable to AM, however, we relax our objective C_1 by defining a new (third) set of distributions for all training samples q_i, $i = 1, \ldots, n$ denoted collectively like the above using the notation $\mathrm{q} \triangleq (q_1, \ldots, q_n)$. We define a new objective to be optimized as follows:

$$\min_{\mathrm{p},\mathrm{q}} C_2(\mathrm{p},\mathrm{q}), \text{ where } C_2(\mathrm{p},\mathrm{q}) = \Bigg[\sum_{i=1}^{l} D_{KL}\big(r_i || q_i\big) + \mu \sum_{i=1}^{n} \sum_{j \in \mathcal{N}(i)} w'_{ij} D_{KL}\big(p_i || q_j\big) - \nu \sum_{i=1}^{n} H(p_i) \Bigg].$$

Before going further, the reader may be wondering at this juncture how might it be desirable for us to have apparently complicated the objective function in an attempt to yield a more computationally and methodologically superior machine learning procedure. This is indeed the case as will be spelled out below. First, in C_2 we have defined a new weight matrix $[W']_{ij} = w'_{ij}$ of the same size as the original where $W' = W + \alpha \mathbf{I}_n$, where $\mathbf{I}_n$ is the $n \times n$ identity matrix, and where $\alpha \geq 0$ is a non-negative constant (this is the optimization related parameter mentioned above). This has the effect that $w'_{ii} \geq w_{ii}$. In the original objective C_1, w_{ii} is irrelevant since $D_{KL}(p||p) = 0$ for all p, but since there are now two distributions for each training point, there should be encouragement for the two to approach each other. Like C_1, the first term of C_2 ensures that the labeled training data is respected and the last term is a smoothness regularizer, but these are done via different sets of distributions, q and p respectively — this choice is what makes possible the relatively simple analytical update equations given below. Next, we see that the two objective functions in fact have identical solutions when the optimization enforces the constraint that p and q are equal:

$$\min_{(\mathrm{p},\mathrm{q}):\mathrm{p}=\mathrm{q}} C_2(\mathrm{p},\mathrm{q}) = \min_{\mathrm{p}} C_1(\mathrm{p}).$$

Indeed, as α gets large, the solutions considered viable are those only where $\mathrm{p} = \mathrm{q}$. We thus have that:

$$\lim_{\alpha\to\infty} \min_{\mathrm{p},\mathrm{q}} C_2(\mathrm{p},\mathrm{q}) = \min_{\mathrm{p}} C_1(\mathrm{p}).$$

Therefore, the two objectives should yield the same solution as long as $\alpha \geq w_{ij}$ for all i, j. A key advantage of this relaxed objective is that it is amenable to alternating minimization, a method to produce a sequence of sets of distributions $(\mathrm{p}^n, \mathrm{q}^n)$ as follows:

$$\mathrm{p}^n = \underset{\mathrm{p}}{\mathrm{argmin}} C_2(\mathrm{p}, \mathrm{q}^{n-1}), \;\; \mathrm{q}^n = \underset{\mathrm{q}}{\mathrm{argmin}} C_2(\mathrm{p}^n, \mathrm{q}).$$

It can be shown (we omit the rather lengthy proof due to space constraints) that the sequence generated using the above minimizations converges to the minimum of $C_2(\mathrm{p},\mathrm{q})$, i.e.,

$$\lim_{n\to\infty} C_2(\mathrm{p}^{(n)}, \mathrm{q}^{(n)}) = \inf_{\mathrm{p},\mathrm{q}} C_2(\mathrm{p},\mathrm{q}),$$

provided we start with a distribution that is initialized properly $q^{(0)}(y) > 0 \;\forall\; y \in \mathrm{Y}$. The update equations for $\mathrm{p}^{(n)}$ and $\mathrm{q}^{(n)}$ are given by

$$p_i^{(n)}(y) = \frac{1}{\mathcal{Z}_i} \exp^{\frac{\beta_i^{(n-1)}(y)}{\gamma_i}},$$

$$q_i^{(n)}(y) = \frac{r_i(y)\delta(i \leq l) + \mu \sum_j w'_{ji} p_j^{(n)}(y)}{\delta(i \leq l) + \mu \sum_j w'_{ji}},$$

where

$$\gamma_i = \nu + \mu \sum_j w'_{ij},$$

$$\beta_i^{(n-1)}(y) = -\nu + \mu \sum_j w'_{ij}(\log q_j^{(n-1)}(y) - 1)$$

and where $\mathcal{Z}_i$ is a normalizing constant to ensure p_i is a valid probability distribution. Note that each iteration of the proposed framework has a closed form solution and is relatively simple to implement, even for very large graphs. Henceforth we refer to the proposed objective optimized using alternating minimization as *AM*.

4 Connections to Other Approaches

Label propagation (LP) (Zhu and Ghahramani, 2002) is a graph-based SSL algorithms that performs Markov random walks on the graph and has a straightforward extension to multi-class problems. The update equations for LP (which also we use for our LP implementations) may be written as

$$p_i^{(n)}(y) = \frac{r_i(y)\delta(i \leq l) + \delta(i > l) \sum_j w_{ij} p_j^{(n-1)}(y)}{\delta(i \leq l) + \delta(i > l) \sum_j w_{ij}}$$

Note the similarity to the update equation for $q_i^{(n)}$ in our AM case. It has been shown that the squared-loss based SSL algorithm (Zhu et al., 2003) and LP have similar updates (Bengio et al., 2007).

The proposed objective C_1 is similar in spirit to the squared-loss based objective in (Zhu et al., 2003; Bengio et al., 2007). Our method, however, differs in that we are optimizing the KL-divergence over probability distributions. We show in section 5 that KL-divergence based loss significantly outperforms the squared-loss. We believe that this could be due to the following: 1) squared loss is appropriate under a Gaussian loss model which may not be optimal under many circumstances (e.g. classification); 2) KL-divergence $D_{KL}(p||q)$ is based on a relative (relative to p) rather than an absolute error; and 3) under certain natural assumptions, KL-divergence is asymptotically consistent with respect to the underlying probability distributions.

AM is also similar to the spectral graph transducer (Joachims, 2003) in that they both attempt to find labellings over the unlabeled data that respect the smoothness constraints of the graph. While spectral graph transduction is an *approximate* solution to a discrete optimization problem (which is NP hard), AM is an *exact* solution obtained by optimizing a convex function over a continuous space. Further, while spectral graph transduction assumes binary classification problems, AM naturally extends to multi-class situations without loss of convexity.

Entropy Minimization (EnM) (Grandvalet and Bengio, 2004) uses the entropy of the unlabeled data as a regularizer while optimizing a parametric loss function defined over the labeled data. While the objectives in the case of both AM and EnM make use of the entropy of the unlabeled data, there are several important differences: (a) EnM is *not* graph-based, (b) EnM is parametric whereas our proposed approach is non-parametric, and most importantly, (c) EnM attempts to *minimize* entropy while the proposed approach aims to *maximize* entropy. While this may seem a triviality, it has catastrophic consequences in terms of both the mathematics and meaning. The objective in case of EnM is not convex, whereas in our case we have a convex formulation with simple update equations and convergence guarantees.

(Wang et al., 2008) is a graph-based SSL algorithm that also employs alternating minimization style optimization. However, it is inherently squared-loss based which our proposed approach out-performs (see section 5). Further, they do not provide or state convergence guarantees and one side of their update approximates an NP-complete optimization procedure.

The information regularization (IR) (Corduneanu and Jaakkola, 2003) algorithm also makes use of a KL-divergence based loss for SSL. Here the input space is divided into regions $\{R_i\}$ which might or might not overlap. For a given point $x_i \in R_i$, IR attempts to minimize the KL-divergence between $p_i(y_i|x_i)$ and $\hat{p}_{R_i}(y)$, the agglomerative distribution for region R_i. Given a graph, one can define a region to be a vertex and its neighbor thus making IR amenable to graph-based SSL. In (Corduneanu and Jaakkola, 2003), the agglomeration is performed by a simple averaging (arithmetic mean). While IR suggests (without proof of convergence) the use of alternating minimization for optimization, one of the steps of the optimization does *not* admit a closed-form solution. This is a serious practical drawback especially in the case of large data sets. (Tsuda, 2005) (hereafter referred to as PD) is an extension of the IR algorithm to hypergraphs where the agglomeration is performed using the geometric mean. This leads to closed form solutions in both steps of the alternating minimization. There are several important differences between IR and PD on one side and our proposed approach: (a) neither IR nor PD use an entropy regularizer, and (b) the update equation for one of the steps of the optimization in the case of PD (equation 13 in (Tsuda, 2005)) is actually a special case of our update equation for $p_i(y)$ and may be obtained by setting $w_{ij} = 1/2$. Further, our work here may be easily extended to hypergraphs.

5 Results

We compare our algorithm (AM) with other state-of-the-art SSL-based text categorization algorithms, namely, (a) SVM (Joachims, 1999), (b) Transductive-SVM (TSVM) (Joachims, 1999), (c) Spectral Graph Transduction (SGT) (Joachims, 2003), and (d) Label Propagation (LP) (Zhu and Ghahramani, 2002). Note that only SGT and LP are graph-based algorithms, while SVM is fully-supervised (i.e., it does not make use of any of the unlabeled data). We implemented SVM and TSVM using *SVM Light* (Joachims, b) and SGT using *SGT Light* (Joachims, a). In the case of SVM, TSVM and SGT we trained |Y| classifiers (one for each class) in a one vs. rest manner precisely following (Joachims, 2003).

5.1 Reuters-21578

We used the "ModApte" split of the Reuters-21578 dataset collected from the Reuters newswire in

1987 (Lewis et al., 1987). The corpus has 9,603 training (not to be confused with $\mathcal{D}$) and 3,299 test documents (which represents $\mathcal{D}_u$). Of the 135 potential topic categories only the 10 most frequent categories are used (Joachims, 1999). Categories outside the 10 most frequent were collapsed into one class and assigned a label "other". For each document i in the training and test sets, we extract features $\mathbf{x}_i$ in the following manner: stop-words are removed followed by the removal of case and information about inflection (i.e., stemming) (Porter, 1980). We then compute TFIDF features for each document (Salton and Buckley, 1987). All graphs were constructed using cosine similarity with TFIDF features.

For this task $\mathrm{Y} = \{$ *earn*, *acq*, *money*, *grain*, *crude*, *trade*, *interest*, *ship*, *wheat*, *corn*, *average*$\}$. For LP and AM, we use the output space $\mathrm{Y}' = \mathrm{Y} \cup \{$ *other* $\}$. For documents in $\mathcal{D}_l$ that are labeled with multiple categories, we initialize r_i to have equal non-zero probability for each such category. For example, if document i is annotated as belonging to classes $\{$ *acq*, *grain*, *wheat*$\}$, then $r_i(\mathit{acq}) = r_i(\mathit{grain}) = r_i(\mathit{wheat}) = 1/3$.

We created 21 transduction sets by randomly sampling l documents from the training set with the constraint that each of 11 categories (top 10 categories and the class *other*) are represented at least once in each set. These samples constitute $\mathcal{D}_l$. All algorithms used the same transduction sets. In the case of SGT, LP and AM, the first transduction set was used to tune the hyperparameters which we then held fixed for all the remaining 20 transduction sets. For all the graph-based approaches, we ran a search over $\mathcal{K} \in \{2, 10, 50, 100, 250, 500, 1000, 2000, n\}$ (note $\mathcal{K} = n$ represents a fully connected graph). In addition, in the case of AM, we set $\alpha = 2$ for all experiments, and we ran a search over $\mu \in \{$1e–8, 1e–4, 0.01, 0.1, 1, 10, 100$\}$ and $\nu \in \{$1e–8, 1e–6, 1e–4, 0.01, 0.1$\}$, for SGT the search was over $c \in \{$3000, 3200, 3400, 3800, 5000, 100000$\}$ (see (Joachims, 2003)).

We report precision-recall break even point (PRBEP) results on the 3,299 test documents in Table 1. PRBEP has been a popular measure in information retrieval (see e.g. (Raghavan et al., 1989)). It is defined as that value for which precision and recall are equal. Results for each category in Table 1 were obtained by averaging the PRBEP over the 20 transduction sets. The final row "average" was obtained by macro-averaging (average of averages). The optimal value of the hyperparameters in case of LP was $\mathcal{K} = 100$; in case of AM, $\mathcal{K} = 2000$, $\mu =$ 1e–4, $\nu =$ 1e–2; and in the case of SGT, $\mathcal{K} = 100$, $c = 3400$. The results show that AM outperforms the state-of-the-art on 6 out of 10 categories and is competitive in 3 of the remaining 4 categories. Further it significantly outperforms all other approaches in case of the macro-averages. AM is significant over its best competitor SGT at the 0.0001 level according to the difference of proportions significance test.

Category	SVM	TSVM	SGT	LP	AM
earn	91.3	95.4	90.4	96.3	**97.9**
acq	67.8	76.6	91.9	90.8	**97.2**
money	41.3	60.0	65.6	57.1	**73.9**
grain	56.2	**68.5**	43.1	33.6	41.3
crude	40.9	**83.6**	65.9	74.8	55.5
trade	29.5	34.0	36.0	**56.0**	47.0
interest	35.6	50.8	50.7	47.9	**78.0**
ship	32.5	46.3	**49.0**	26.4	39.6
wheat	47.9	44.4	59.1	58.2	**64.3**
corn	41.3	33.7	51.2	55.9	**68.3**
average	48.9	59.3	60.3	59.7	**66.3**

Table 1: P/R Break Even Points (PRBEP) for the top 10 categories in the Reuters data set with $l = 20$ and $u = 3299$. All results are averages over 20 randomly generated transduction sets. The last row is the macro-average over all the categories. Note AM is the proposed approach.

Figure 1 shows the variation of "average" PRBEP against the number of labeled documents (l). For each value of l, we tuned the hyperparameters over the first transduction set and used these values for all the other 20 sets. Figure 1 also shows error-bars ($\pm$ standard deviation) all the experiments. As expected, the performance of all the approaches improves with increasing number of labeled documents. Once again in this case, AM, outperforms the other approaches for all values of l.

5.2 WebKB Collection

World Wide Knowledge Base (WebKB) is a collection of 8282 web pages obtained from four academic

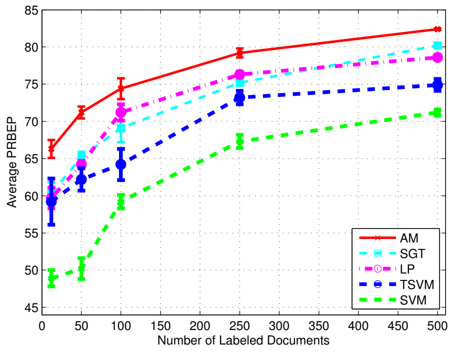


Figure 1: Average PRBEP over all classes vs. number of labeled documents (l) for Reuters data set

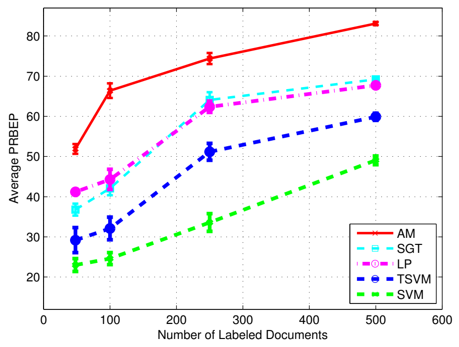


Figure 2: Average PRBEP over all classes vs. number of labeled documents (l) for WebKB collection.

domains. The web pages in the WebKB set are labeled using two different polychotomies. The first is according to topic and the second is according to web domain. In our experiments we only considered the first polychotomy, which consists of 7 categories: *course*, *department*, *faculty*, *project*, *staff*, *student*, and *other*. Following (Nigam et al., 1998) we only use documents from categories *course*, *department*, *faculty*, *project* which gives 4199 documents for the four categories. Each of the documents is in HTML format containing text as well as other information such as HTML tags, links, etc. We used both textual and non-textual information to construct the feature vectors. In this case we did not use either stop-word removal or stemming as this has been found to hurt performance on this task (Nigam et al., 1998). As in the the case of the Reuters data set we extracted TFIDF features for each document and constructed the graph using cosine similarity.

As in (Bekkerman et al., 2003), we created four roughly-equal random partitions of the data set. In order to obtain $\mathcal{D}_l$, we first randomly choose a split and then sample l documents from that split. The other three splits constitute $\mathcal{D}_u$. We believe this is more realistic than sampling the labeled web-pages from a single university and testing web-pages from the other universities (Joachims, 1999). This method of creating transduction sets allows us to better evaluate the generalization performance of the various algorithms. Once again we create 21 transduction sets and the first set was used to tune the hyperparameters. Further, we ran a search over the same grid as used in the case of Reuters. We report precision-recall break even point (PRBEP) results on the 3,148 test documents in Table 2. For this task, we found that the optimal value of the hyperparameter were: in the case of LP, $\mathcal{K} = 1000$; in case of AM, $\mathcal{K} = 1000$, $\mu = 1\text{e–}2$, $\nu = 1\text{e–}4$; and in case of SGT, $\mathcal{K} = 100$, $c = 3200$. Once again, AM is significant at the 0.0001 level over its closest competitor LP. Figure 2 shows the variation of PRBEP with number of labeled documents (l) and was generated in a similar fashion as in the case of the Reuters data set.

Class	SVM	TSVM	SGT	LP	AM
course	46.5	43.9	29.9	45.0	**67.6**
faculty	14.5	31.2	**42.9**	40.3	42.5
project	15.8	17.2	17.5	27.8	**42.3**
student	15.0	24.5	56.6	51.8	**55.0**
average	23.0	29.2	36.8	41.2	**51.9**

Table 2: P/R Break Even Points (PRBEP) for the WebKB data set with $l = 48$ and $u = 3148$. All results are averages over 20 randomly generated transduction sets. The last row is the macro-average over all the classes. AM is the proposed approach.

6 Discussion

We note that LP may be cast into an AM-like framework by using the following sequence of updates,

$$p_i^{(n)}(y) = \delta(i \le l)r_i(y) + \delta(i > l)q_i^{(n-1)},$$
$$q_i^{(n)}(y) = \frac{\sum_j w_{ij} p_i^{(n)}(y)}{\sum_j w_{ij}}$$

To compare the behavior of AM and LP, we applied this form of LP along with AM on a simple 5-node binary-classification SSL graph where two nodes are labeled (node 1 and 2) and the remaining nodes are unlabeled (see Figure 3, top). Since this is binary classification ($|Y| = 2$), each distribution p_i or q_i can be depicted using only a single real number between 0 and 1 corresponding to the probability that each vertex is class 2 (yes two). We show how both LP and AM evolve starting from exactly the same random starting point q^0 (Figure 3, bottom). For each algorithm, the figure shows that both algorithms clearly converge. Each alternate iteration of LP is such that the labeled vertices oscillate due to its clamping back to the labeled distribution, but that is not the case for AM. We see, moreover, qualitative differences in the solutions as well – e.g., AM's solution for the pendant node 5 is less confident than is LP's solution. More empirical comparative analysis between the two algorithms of this sort will appear in future work.

We have proposed a new algorithm for semi-supervised text categorization. Empirical results show that the proposed approach significantly outperforms the state-of-the-art. In addition the proposed approach is relatively simple to implement and has guaranteed convergence properties. While in this work, we use relatively simple features to construct the graph, use of more sophisticated features and/or similarity measures could lead to further improved results.

Acknowledgments

This work was supported by ONR MURI grant N000140510388, by NSF grant IIS-0093430, by the Companions project (IST programme under EC grant IST-FP6-034434), and by a Microsoft Research Fellowship.

References

Alexandrescu, A. and Kirchhoff, K. (2007). Data-driven graph construction for semi-supervised graph-based learnning in nlp. In *Proc. of the Human Language Technologies Conference (HLT-NAACL)*.

Bekkerman, R., El-Yaniv, R., Tishby, N., and Winter, Y. (2003). Distributional word clusters vs. words for text categorization. *J. Mach. Learn. Res.*, 3:1183–1208.

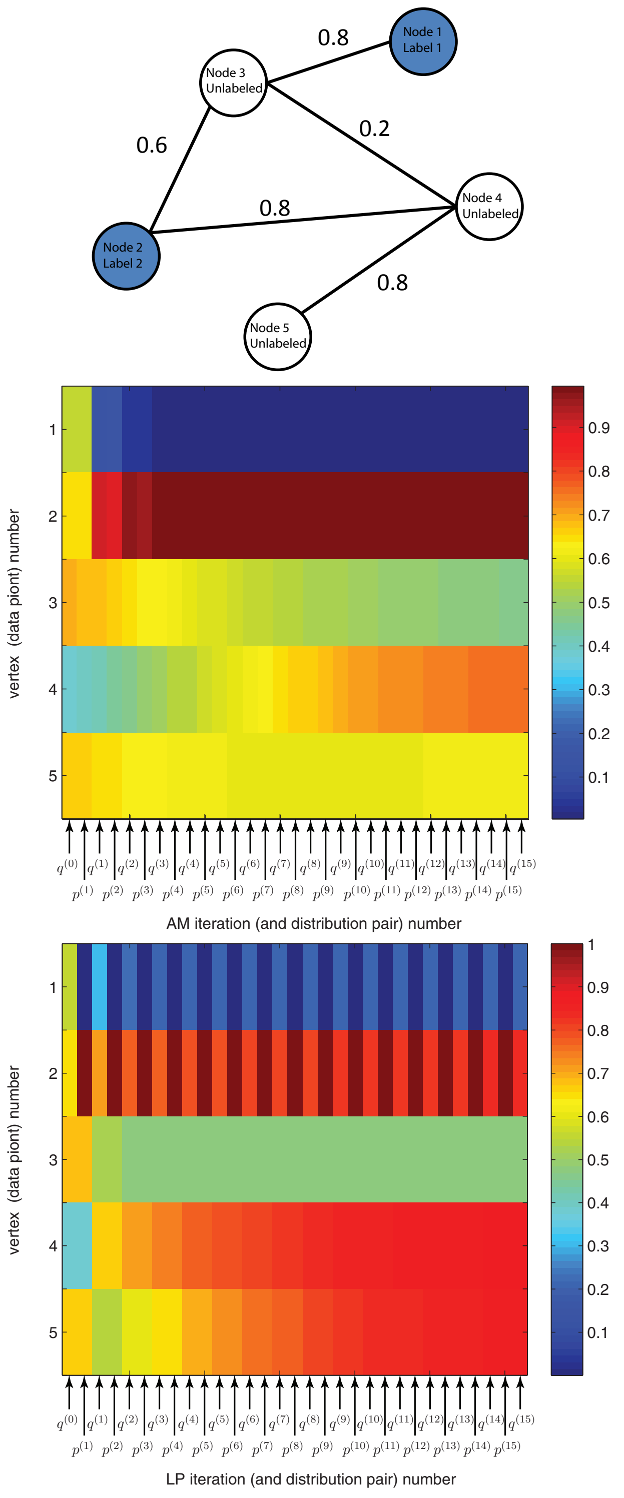


Figure 3: Graph (top), and alternating values of $\mathrm{p}^n, \mathrm{q}^n$ for increasing n for AM and LP.

Belkin, M., Niyogi, P., and Sindhwani, V. (2005). On manifold regularization. In *Proc. of the Conference on Artificial Intelligence and Statistics (AISTATS)*.

Bengio, Y., Delalleau, O., and Roux, N. L. (2007). *Semi-Supervised Learning*, chapter Label Propogation and Quadratic Criterion. MIT Press.

Bertsekas, D. (2004). *Nonlinear Programming*. Athena Scientific Publishing.

Blitzer, J. and Zhu, J. (2008). ACL 2008 tutorial on Semi-Supervised learning. `http://ssl-acl08.wikidot.com/`.

Blum, A. and Chawla, S. (2001). Learning from labeled and unlabeled data using graph mincuts. In *Proc. 18th International Conf. on Machine Learning*, pages 19–26. Morgan Kaufmann, San Francisco, CA.

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory*.

Chapelle, O., Scholkopf, B., and Zien, A. (2007). *Semi-Supervised Learning*. MIT Press.

Corduneanu, A. and Jaakkola, T. (2003). On information regularization. In *Uncertainty in Artificial Intelligence*.

Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory*. Wiley Series in Telecommunications. Wiley, New York.

Csiszar, I. and Tusnady, G. (1984). Information Geometry and Alternating Minimization Procedures. *Statistics and Decisions*.

Dumais, S., Platt, J., Heckerman, D., and Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *CIKM '98: Proceedings of the seventh international conference on Information and knowledge management*, New York, NY, USA.

Grandvalet, Y. and Bengio, Y. (2004). Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems (NIPS)*.

Joachims, T. SGT Light. `http://sgt.joachims.org`.

Joachims, T. SVM Light. `http://svmlight.joachims.org`.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Proc. of the International Conference on Machine Learning (ICML)*.

Joachims, T. (2003). Transductive learning via spectral graph partitioning. In *Proc. of the International Conference on Machine Learning (ICML)*.

Lewis, D. et al. (1987). Reuters-21578. `http://www.daviddlewis.com/resources/testcollections/reuters21578`.

Nigam, K., McCallum, A., Thrun, S., and Mitchell, T. (1998). Learning to classify text from labeled and unlabeled documents. In *AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pages 792–799.

Pearl, J. (1990). *Jeffrey's Rule, Passage of Experience and Neo-Bayesianism in Knowledge Representation and Defeasible Reasoning*. Kluwer Academic Publishers.

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

Raghavan, V., Bollmann, P., and Jung, G. S. (1989). A critical investigation of recall and precision as measures of retrieval system performance. *ACM Trans. Inf. Syst.*, 7(3):205–229.

Salton, G. and Buckley, C. (1987). Term weighting approaches in automatic text retrieval. Technical report, Ithaca, NY, USA.

Sindhwani, V., Niyogi, P., and Belkin, M. (2005). Beyond the point cloud: from transductive to semi-supervised learning. In *Proc. of the International Conference on Machine Learning (ICML)*.

Szummer, M. and Jaakkola, T. (2001). Partially labeled classification with Markov random walks. In *Advances in Neural Information Processing Systems*, volume 14.

Tsuda, K. (2005). Propagating distributions on a hypergraph by dual information regularization. In *Proceedings of the 22nd International Conference on Machine Learning*.

Wang, J., Jebara, T., and Chang, S.-F. (2008). Graph transduction via alternating minimization. In *Proc. of the International Conference on Machine Learning (ICML)*.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*.

Zhu, X. (2005a). Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison.

Zhu, X. (2005b). *Semi-Supervised Learning with Graphs*. PhD thesis, Carnegie Mellon University.

Zhu, X. and Ghahramani, Z. (2002). Learning from labeled and unlabeled data with label propagation. Technical report, Carnegie Mellon University.

Zhu, X., Ghahramani, Z., and Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *Proc. of the International Conference on Machine Learning (ICML)*.

Author Index