

ACL 2014

**52nd Annual Meeting of the
Association for Computational Linguistics**

TACL Papers

June 23-25, 2014
Baltimore, Maryland, USA

©2014 The Association for Computational Linguistics

Order copies of this and other ACL proceedings from:

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

Table of Contents

<i>Training Deterministic Parsers with Non-Deterministic Oracles</i> Yoav Goldberg and Joakim Nivre	1
<i>Joint Incremental Disfluency Detection and Dependency Parsin</i> Matthew Honnibal and Mark Johnson	13
<i>A Crossing-Sensitive Third-Order Factorization for Dependency Parsing</i> Emily Pitler	25
<i>Exploring the Role of Stress in Bayesian Word Segmentation using Adaptor Grammars</i> Benjamin Börschinger and Mark Johnson	39
<i>FLORS: Fast and Simple Domain Adaptation for Part-of-Speech Tagging</i> Tobias Schnabel and Hinrich Schütze	51
<i>A Tabular Method for Dynamic Oracles in Transition-Based Parsing</i> Yoav Goldberg, Francesco Sartorio and Giorgio Satta	63
<i>Temporal Annotation in the Clinical Domain</i> William Styler, Steven Bethard, Sean Finan, Martha Palmer, Sameer Pradhan, Piet C de Groen, Brad Erickson, Timothy Miller, Chen Lin, Guergana Savova and James Pustejovsky	75
<i>Entity Linking meets Word Sense Disambiguation: a Unified Approach</i> Andrea Moro, Alessandro Raganato and Roberto Navigli	87
<i>Data-Driven Metaphor Recognition and Explanation</i> Hongsong Li, Kenny Q. Zhu and Haixun Wang	101
<i>Grounded Compositional Semantics for Finding and Describing Images with Sentences</i> Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher Manning and Andrew Ng	113
<i>Parallel Algorithms for Unsupervised Tagging</i> Sujith Ravi, Sergei Vassilivitskii and Vibhor Rastogi	125
<i>Heterogeneous Networks and Their Applications: Scientometrics, Name Disambiguation, and Topic Modeling</i> Ben King, Rahul Jha and Dragomir R. Radev	139
<i>Discriminative Lexical Semantic Segmentation with Gaps: Running the MWE Gamut</i> Nathan Schneider, Emily Danchik, Chris Dyer and Noah A. Smith	153
<i>Segmentation for Efficient Supervised Language Annotation with an Explicit Cost-Utility Tradeoff</i> Matthias Sperber, Mirjam Simantzik, Graham Neubig, Satoshi Nakamura and Alex Waibel ...	167
<i>The Language Demographics of Amazon Mechanical Turk</i> Ellie Pavlick, Matt Post, Ann Irvine, Dmitry Kachaev and Chris Callison-Burch	179
<i>Cross-lingual Projected Expectation Regularization for Weakly Supervised Learning</i> Mengqiu Wang and Christopher Manning	193
<i>Back to Basics for Monolingual Alignment: Exploiting Word Similarity and Contextual Evidence</i> Md Arafat Sultan, Steven Bethard and Tamara Sumner	205

From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions

Peter Young, Alice Lai, Micah Hodosh and Julia Hockenmaier 217

Senti-LSSVM: Sentiment-Oriented Multi-Relation Extraction with Latent Structural SVM

Lizhen Qu, Yi Zhang, Rui Wang, Lili Jiang, Rainer Gemulla and Gerhard Weikum 229

Conference Program

18:00–21:00 Welcome Reception

Monday, June 23, 2014

7:30–18:00 Registration

7:30–9:00 Breakfast

8:55–9:00 Opening session

9:00–9:40 President talk

9:40–10:10 Coffee break

Session 1A: Discourse, Dialogue, Coreference and Pragmatics

Session 1B: Semantics I

Session 1C: Machine Translation I

Session 1D: Syntax, Parsing, and Tagging I

10:10–10:35 *Training Deterministic Parsers with Non-Deterministic Oracles*
Yoav Goldberg and Joakim Nivre

10:35–11:00 *Joint Incremental Disfluency Detection and Dependency Parsin*
Matthew Honnibal and Mark Johnson

11:25–11:50 *A Crossing-Sensitive Third-Order Factorization for Dependency Parsing*
Emily Pitler

Monday, June 23, 2014 (continued)

Session 1E: NLP for the Web and Social Media I

11:50–13:20 Lunch break; Student Lunch

Session 2A: Syntax, Parsing and Tagging II

Session 2B: Semantics II

Session 2C: Word Segmentation and POS Tagging

13:30–13:45 *Exploring the Role of Stress in Bayesian Word Segmentation using Adaptor Grammars*
Benjamin Börschinger and Mark Johnson

14:10–14:35 *FLORS: Fast and Simple Domain Adaptation for Part-of-Speech Tagging*
Tobias Schnabel and Hinrich Schütze

Session 2D: SRW

Session 2E: Sentiment Analysis I

15:00–15:30 Coffee break

Session 3A: Topic Modeling

Session 3B: Information Extraction I

Monday, June 23, 2014 (continued)

Session 3C: Generation

Session 3D: Syntax, Parsing and Tagging III

15:30–15:55 *A Tabular Method for Dynamic Oracles in Transition-Based Parsing*
Yoav Goldberg, Francesco Sartorio and Giorgio Satta

Session 3E: Language Resources and Evaluation I

16:20–16:45 *Temporal Annotation in the Clinical Domain*
William Styler, Steven Bethard, Sean Finan, Martha Palmer, Sameer Pradhan, Piet C de Groen, Brad Erickson, Timothy Miller, Chen Lin, Guergana Savova and James Pustejovsky

16:45–17:00 Break

17:00–18:00 Invited talk I: Corinna Cortes

Oral Sessions for Student Research Workshop Posters

18:50–21:30 Poster and Dinner Session I: TACL Papers, Long Papers, Short Papers, Student Research Workshop; Demonstrations

Entity Linking meets Word Sense Disambiguation: a Unified Approach
Andrea Moro, Alessandro Raganato and Roberto Navigli

Data-Driven Metaphor Recognition and Explanation
Hongsong Li, Kenny Q. Zhu and Haixun Wang

Grounded Compositional Semantics for Finding and Describing Images with Sentences
Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher Manning and Andrew Ng

Parallel Algorithms for Unsupervised Tagging
Sujith Ravi, Sergei Vassilivitskii and Vibhor Rastogi

Heterogeneous Networks and Their Applications: Scientometrics, Name Disambiguation, and Topic Modeling
Ben King, Rahul Jha and Dragomir R. Radev

Discriminative Lexical Semantic Segmentation with Gaps: Running the MWE Gamut
Nathan Schneider, Emily Danchik, Chris Dyer and Noah A. Smith

Tuesday, June 24, 2014

7:30–18:00 Registration

7:30–9:00 Breakfast

9:00–10:00 Invited talk II: Zoran Popovic

10:00–10:30 Coffee break

Session 4A: Machine Learning for NLP

Session 4B: Information Extraction II

Session 4C: Machine Translation II

Session 4D: Summarization

Session 4E: Language Resources and Evaluation II

10:30–10:55 *Segmentation for Efficient Supervised Language Annotation with an Explicit Cost-Utility Tradeoff*

Matthias Sperber, Mirjam Simantzik, Graham Neubig, Satoshi Nakamura and Alex Waibel

11:45–12:10 *The Language Demographics of Amazon Mechanical Turk*

Ellie Pavlick, Matt Post, Ann Irvine, Dmitry Kachaev and Chris Callison-Burch

12:10–13:30 Lunch break

Tuesday, June 24, 2014 (continued)

Session 5A: Question Answering

Session 5B: Information Extraction III

Session 5C: Lexical Semantics and Ontology I

Session 5D: Syntax, Parsing and Tagging IV

14:20–14:45 *Cross-lingual Projected Expectation Regularization for Weakly Supervised Learning*
Mengqiu Wang and Christopher Manning

Session 5E: Cognitive Modeling and Psycholinguistics

14:45–15:15 Coffee break

Session 6A: Machine Translation III

Session 6B: Lexical Semantics and Ontology II

Session 6C: Generation/Summarization/Dialogue

Session 6D: NLP Applications and NLP Enabled Technology I

Session 6E: Language Resources and Evaluation III

16:50–19:20 Poster and Dinner Session II: Long Papers, Short Papers and Demonstrations in Grand Ballroom I/II/III/IV/V/VI/VII/VIII

19:30–22:00 Social at the National Aquarium in Baltimore

Wednesday, June 25, 2014

7:30–18:00 Registration

7:30–9:00 Breakfast

Best paper session

10:15–10:45 Coffee break

Session 7A: Multimodal NLP/ Lexical Semantics

10:45–11:10 *Back to Basics for Monolingual Alignment: Exploiting Word Similarity and Contextual Evidence*

Md Arafat Sultan, Steven Bethard and Tamara Sumner

12:00–12:25 *From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions*

Peter Young, Alice Lai, Micah Hodosh and Julia Hockenmaier

Session 7B: Semantics III

Session 7C: Machine Translation IV

Session 7D: NLP Applications and NLP Enabled Technology II

Session 7E: Sentiment Analysis II

10:45–11:10 *Senti-LSSVM: Sentiment-Oriented Multi-Relation Extraction with Latent Structural SVM*

Lizhen Qu, Yi Zhang, Rui Wang, Lili Jiang, Rainer Gemulla and Gerhard Weikum

12:25–13:30 Lunch break

13:30–15:00 ACL Business Meeting

Wednesday, June 25, 2014 (continued)

Session 8A: NLP for the Web and Social Media II

Session 8B: Semantics/Information Extraction

Session 8C: Machine Translation V

Session 8D: Syntax, Parsing, and Tagging V

Session 8E: Multilinguality and Multimodal NLP

16:30–17:00 Coffee break

17:00–18:30 Lifetime Achievement Award

18:30–19:00 Closing session

Training Deterministic Parsers with Non-Deterministic Oracles

Yoav Goldberg

Bar-Ilan University
Department of Computer Science
Ramat-Gan, Israel
yoav.goldberg@gmail.com

Joakim Nivre

Uppsala University
Department of Linguistics and Philology
Uppsala, Sweden
joakim.nivre@lingfil.uu.se

Abstract

Greedy transition-based parsers are very fast but tend to suffer from error propagation. This problem is aggravated by the fact that they are normally trained using oracles that are deterministic and incomplete in the sense that they assume a unique canonical path through the transition system and are only valid as long as the parser does not stray from this path. In this paper, we give a general characterization of oracles that are nondeterministic and complete, present a method for deriving such oracles for transition systems that satisfy a property we call arc decomposition, and instantiate this method for three well-known transition systems from the literature. We say that these oracles are dynamic, because they allow us to dynamically explore alternative and non-optimal paths during training – in contrast to oracles that statically assume a unique optimal path. Experimental evaluation on a wide range of data sets clearly shows that using dynamic oracles to train greedy parsers gives substantial improvements in accuracy. Moreover, this improvement comes at no cost in terms of efficiency, unlike other techniques like beam search.

1 Introduction

Greedy transition-based parsers are easy to implement and are very efficient, but they are generally not as accurate as parsers that are based on global search (McDonald et al., 2005; Koo and Collins, 2010) or as transition-based parsers that use beam search (Zhang and Clark, 2008) or dynamic programming (Huang and Sagae, 2010; Kuhlmann et

al., 2011). This work is part of a line of research trying to push the boundaries of greedy parsing and narrow the accuracy gap of 2–3% between search-based and greedy parsers, while maintaining the efficiency and incremental nature of greedy parsers.

One reason for the lower accuracy of greedy parsers is error propagation: once the parser makes an error in decoding, more errors are likely to follow. This behavior is closely related to the way in which greedy parsers are normally trained. Given a treebank oracle, a gold sequence of transitions is derived, and a predictor is trained to predict transitions along this gold sequence, without considering any parser state outside this sequence. Thus, once the parser strays from the golden path at test time, it ventures into unknown territory and is forced to react to situations it has never been trained for.

In recent work (Goldberg and Nivre, 2012), we introduced the concept of a *dynamic* oracle, which is non-deterministic and not restricted to a single golden path, but instead provides optimal predictions for any possible state the parser might be in. Dynamic oracles are non-deterministic in the sense that they return a *set* of valid transitions for a given parser state and gold tree. Moreover, they are well-defined and optimal also for states from which the gold tree cannot be derived, in the sense that they return the set of transitions leading to the best tree derivable from each state. We showed experimentally that, using a dynamic oracle for the arc-eager transition system (Nivre, 2003), a greedy parser can be trained to perform well also after incurring a mistake, thus alleviating the effect of error propagation and resulting in consistently better parsing accuracy.

In this paper, we extend the work of Goldberg and Nivre (2012) by giving a general characterization of dynamic oracles as oracles that are *non-deterministic*, in that they return sets of transitions, and *complete*, in that they are defined for all possible states. We then define a formal property of transition systems which we call *arc decomposition*, and introduce a framework for deriving dynamic oracles for arc-decomposable systems. Using this framework, we derive novel dynamic oracles for the hybrid (Kuhlmann et al., 2011) and easy-first (Goldberg and Elhadad, 2010) transition systems, which are arc-decomposable (as is the arc-eager system). We also show that the popular arc-standard system (Nivre, 2004) is not arc-decomposable, and so deriving a dynamic oracle for it remains an open research question. Finally, we perform a set of experiments on the CoNLL 2007 data sets, validating that the use of dynamic oracles for exploring states that result from parsing mistakes during training is beneficial across transition systems.

2 Transition-Based Dependency Parsing

We begin with a quick review of transition-based dependency parsing, presenting the arc-eager, arc-standard, hybrid and easy-first transitions systems in a common notation. The transition-based parsing framework (Nivre, 2008) assumes a *transition system*, an abstract machine that processes sentences and produces parse trees. The transition system has a set of *configurations* and a set of *transitions* which are applied to configurations. When parsing a sentence, the system is initialized to an *initial configuration* based on the input sentence, and transitions are repeatedly applied to this configuration. After a finite number of transitions, the system arrives at a *terminal configuration*, and a parse tree is read off the terminal configuration. In a greedy parser, a classifier is used to choose the transition to take in each configuration, based on features extracted from the configuration itself. Transition systems differ by the way they define configurations, and by the particular set of transitions available.

2.1 Dependency Trees

We define a *dependency tree* for a sentence $W = w_1, \dots, w_n$ to be a labeled directed tree $T = (V, A)$,

where $V = \{w_1, \dots, w_n\}$ is a set of nodes given by the tokens of the input sentence, and $A \subseteq V \times L \times V$ (for some dependency label set L) is a set of labeled directed arcs of the form (h, lb, d) , where $h \in V$ is said to be the head, $d \in V$ the dependent, and $lb \in L$ the dependency label.

When dealing with unlabeled parsing, or when the label identity is irrelevant, we take $A \subseteq V \times V$ to be a set of ordinary directed arcs of the form (h, d) . Note that, since the nodes of the tree are given by the input sentence, a dependency tree $T = (V, A)$ for a sentence W is uniquely defined by the arc set A . For convenience, we will therefore equate the tree with the arc set and use the symbol T for the latter, reserving the symbol A for arc sets that are not necessarily trees. In the context of this work it is assumed that all the dependency trees are *projective*.

Although the general definition of a dependency tree does not make any assumptions about which node is the root of the tree, it is common practice in dependency parsing to add a dummy node ROOT, which is prefixed or suffixed to the sentence and which always acts as the root of the tree. We will follow this practice in our description of different transition systems below.

2.2 Transition Systems

Arc-Eager In the arc-eager system (Nivre, 2003), a configuration $c = (\sigma, \beta, A)$ consists of a stack σ , a buffer β , and a set A of dependency arcs.¹ Given a sentence $W = w_1, \dots, w_n$, the system is initialized with an empty stack, an empty arc set, and $\beta = w_1, \dots, w_n, \text{ROOT}$, where ROOT is the special root node. Any configuration c with an empty stack and a buffer containing only ROOT is terminal, and the parse tree is given by the arc set A_c of c .² The system has 4 transitions: RIGHT_{lb}, LEFT_{lb}, SHIFT, REDUCE, defined as follows:

$$\begin{aligned} \text{SHIFT}[(\sigma, b|\beta, A)] &= (\sigma|b, \beta, A) \\ \text{RIGHT}_{lb}[(\sigma|s, b|\beta, A)] &= (\sigma|s|b, \beta, A \cup \{(s, lb, b)\}) \\ \text{LEFT}_{lb}[(\sigma|s, b|\beta, A)] &= (\sigma, b|\beta, A \cup \{(b, lb, s)\}) \\ \text{REDUCE}[(\sigma|s, \beta, A)] &= (\sigma, \beta, A) \end{aligned}$$

¹We use $\sigma|x$ to denote a stack with top element x and remainder σ , and $x|\beta$ to denote a buffer with a head x followed by the elements in β .

²This definition of a terminal configuration differs from that in Nivre (2003) but guarantees that the set A_c is a dependency tree rooted in ROOT.

There is a precondition on the RIGHT and SHIFT transitions to be legal only when $b \neq \text{ROOT}$, and for LEFT, RIGHT and REDUCE to be legal only when the stack is non-empty. Moreover, LEFT is only legal when s does not have a parent in A , and REDUCE when s does have a parent in A . In general, we use $\text{LEGAL}(c)$ to refer to the set of transitions that are legal in a configuration c . The arc-eager system builds trees *eagerly* in the sense that arcs are added at the earliest time possible. In addition, each word will collect all of its left dependents before collecting its right dependents.

Arc-Standard The arc-standard system (Nivre, 2004) has configurations of the same form $c = (\sigma, \beta, A)$ as the arc-eager system. The initial configuration for a sentence $W = w_1, \dots, w_n$ has an empty stack and arc set and $\beta = \text{ROOT}, w_1, \dots, w_n$. A configuration c is terminal if it has an empty buffer and a stack containing the single node ROOT ; the parse tree is given by A_c . The system has 3 transitions: RIGHT_{lb} , LEFT_{lb} , SHIFT , defined as follows:

$$\begin{aligned} \text{SHIFT}[(\sigma, b|\beta, A)] &= (\sigma|b, \beta, A) \\ \text{RIGHT}_{lb}[(\sigma|s_1|s_0, \beta, A)] &= (\sigma|s_1, \beta, A \cup \{(s_1, lb, s_0)\}) \\ \text{LEFT}_{lb}[(\sigma|s_1|s_0, \beta, A)] &= (\sigma|s_0, \beta, A \cup \{(s_0, lb, s_1)\}) \end{aligned}$$

There is a precondition on the LEFT transition to be legal only when $s_1 \neq \text{ROOT}$, and for LEFT and RIGHT to be legal only when the stack has at least two elements. The arc-standard system builds trees in a *bottom-up* fashion: each word must collect all its dependents before being attached to its head. The system does not pose any restriction with regard to the order of collecting left and right dependents.

Hybrid The hybrid system (Kuhlmann et al., 2011) has the same configurations and the same initialization and termination conditions as the arc-standard system. The system has 3 transitions: RIGHT_{lb} , LEFT_{lb} , SHIFT , defined as follows:

$$\begin{aligned} \text{SHIFT}[(\sigma, b|\beta, A)] &= (\sigma|b, \beta, A) \\ \text{RIGHT}_{lb}[(\sigma|s_1|s_0, \beta, A)] &= (\sigma|s_1, \beta, A \cup \{(s_1, lb, s_0)\}) \\ \text{LEFT}_{lb}[(\sigma|s, b|\beta, A)] &= (\sigma, b|\beta, A \cup \{(b, lb, s)\}) \end{aligned}$$

There is a precondition on RIGHT to be legal only when the stack has at least two elements, and on LEFT to be legal only when the stack is non-empty and $s \neq \text{ROOT}$. The hybrid system can be seen as a combination of the arc-standard and arc-eager

Algorithm 1 Greedy transition-based parsing

- 1: **Input:** sentence W , parameter-vector \mathbf{w}
 - 2: $c \leftarrow \text{INITIAL}(W)$
 - 3: **while not** $\text{TERMINAL}(c)$ **do**
 - 4: $t_p \leftarrow \arg \max_{t \in \text{LEGAL}(c)} \mathbf{w} \cdot \phi(c, t)$
 - 5: $c \leftarrow t_p(c)$
 - 6: **return** A_c
-

systems, using the LEFT action of arc-eager and the RIGHT action of arc-standard. Like arc-standard, it builds trees in a bottom-up fashion. But like arc-eager, it requires a word to collect all its left dependents before collecting any right dependent.

Easy-First In the easy-first system (Goldberg and Elhadad, 2010), a configuration $c = (\lambda, A)$ consists of a list λ and a set A of dependency arcs. We use l_i to denote the i th member of λ and write $|\lambda|$ for the length of λ . Given a sentence $W = w_1, \dots, w_n$, the system is initialized with an empty arc set and $\lambda = \text{ROOT}, w_1, \dots, w_n$. A configuration c is terminal with parse tree A_c if $\lambda = \text{ROOT}$. The set of transitions for a given configuration $c = (\lambda, A)$ is:

$$\begin{aligned} &\{\text{LEFT}_{lb}^i | 1 < i \leq |\lambda|\} \cup \{\text{RIGHT}_{lb}^i | 1 \leq i < |\lambda|\}, \text{ where:} \\ \text{LEFT}_{lb}^i[(\lambda, A)] &= (\lambda \setminus \{l_{i-1}\}, A \cup \{(l_i, lb, l_{i-1})\}) \\ \text{RIGHT}_{lb}^i[(\lambda, A)] &= (\lambda \setminus \{l_{i+1}\}, A \cup \{(l_i, lb, l_{i+1})\}) \end{aligned}$$

There is a precondition on LEFT_{lb}^i transitions to only trigger if $l_{i-1} \neq \text{ROOT}$. Unlike the arc-eager, arc-standard and hybrid transition systems that work in a *left-to-right* order and access the sentence incrementally, the easy-first system is *non-directional* and has access to the entire sentence at each step. Like the arc-standard and hybrid systems, it builds trees bottom-up.

2.3 Greedy Transition-Based Parsing

Assuming that we have a feature-extraction function $\phi(c, t)$ over configurations c and transitions t and a weight-vector \mathbf{w} assigning weights to each feature, greedy transition-based parsing is very simple and efficient using Algorithm 1. Starting in the initial configuration for a given sentence, we repeatedly choose the highest-scoring transition according to our model and apply it, until we reach a terminal configuration, at which point we stop and return the parse tree accumulated in the configuration.

Algorithm 2 Online training of greedy transition-based parsers (i th iteration)

```
1: for sentence  $W$  with gold tree  $T$  in corpus do
2:    $c \leftarrow \text{INITIAL}(W)$ 
3:   while not  $\text{TERMINAL}(c)$  do
4:      $\text{CORRECT}(c) \leftarrow \{t \mid o(t; c, T) = \text{true}\}$ 
5:      $t_p \leftarrow \arg \max_{t \in \text{LEGAL}(c)} \mathbf{w} \cdot \phi(c, t)$ 
6:      $t_o \leftarrow \arg \max_{t \in \text{CORRECT}(c)} \mathbf{w} \cdot \phi(c, t)$ 
7:     if  $t_p \notin \text{CORRECT}(c)$  then
8:        $\text{UPDATE}(\mathbf{w}, \phi(c, t_o), \phi(c, t_p))$ 
9:        $c \leftarrow \text{NEXT}(c, t_o)$ 
10:    else
11:       $c \leftarrow t_p(c)$ 
```

3 Training Transition-Based Parsers

We now turn to the training of greedy transition-based parsers, starting with a review of the standard method using static oracles and moving on to the idea of training with exploration proposed by Goldberg and Nivre (2012).

3.1 Training with Static Oracles

The standard approach to training greedy transition-based parsers is illustrated in Algorithm 2.³ It assumes the existence of an *oracle* $o(t; c, T)$, which returns **true** if transition t is correct for configuration c and gold tree T . Given this oracle, training is very similar to parsing, but after predicting the next transition t_p using the model in line 5 we check if it is contained in the set $\text{CORRECT}(c)$ of transitions that are considered correct by the oracle (lines 4 and 7). If the predicted transition t_p is not correct, we update the model parameters \mathbf{w} away from t_p and toward the oracle prediction t_o , which is the highest-scoring correct transition under the current model, and move on to the next configuration (lines 7–9). If t_p is correct, we simply apply it and move to $t_p(c)$ without changing the model parameters (line 11).

The function $\text{NEXT}(c, t_o)$ in line 9 is used to abstract over a subtle difference in the standard training procedure for the left-to-right systems (arc-eager, arc-standard and hybrid), on the one hand,

³We present the standard approach as an online algorithm in order to ease the transition to the novel approach. While some transition-based parsers use batch learning instead, the essential point is that they explore exactly the same configurations during the training phase.

and the easy-first system, on the other. In the former case, $\text{NEXT}(c, t_o)$ evaluates to $t_o(c)$, which means that we apply the oracle transition t_o and move on to the next configuration. For the easy-first system, $\text{NEXT}(c, t_o)$ instead evaluates to c , which means that we remain in the same configuration for as many updates as necessary to get a correct model prediction.

Traditionally, the oracles for the left-to-right systems are *static*: they return a single correct transition and are only correct for configurations that result from transitions predicted by the oracle itself. The oracle for the easy-first system is *non-deterministic* and returns a set of correct transitions. However, like the static oracle, it is correct only for configurations from which the gold tree is reachable. Thus, in both cases, we need to make sure that a transition is applied during training only if it is considered correct by the oracle; else we cannot guarantee that later oracle predictions will be correct. Therefore, on line 9, we either remain in the same configuration (easy-first) or follow the oracle prediction and go to $t_o(c)$ (left-to-right systems); on line 11, we in fact also go to $t_o(c)$, because in this case we have $t_p(c) = t_o(c)$.

A notable shortcoming of this training procedure is that, at parsing time, the parsing model may predict incorrect transitions and reach configurations that are not on the oracle path. Since the model has never seen such configurations during training, it is likely to perform badly in them, making further mistakes more likely. We would therefore like the parser to encounter configurations resulting from incorrect transitions during training and learn what constitutes optimal transitions in such configurations. Unfortunately, this is not possible using the static (or even the non-deterministic) oracles.

3.2 Training with Exploration

Assuming we had access to an oracle that could tell us which transitions are *optimal* in any configuration, including ones from which the gold tree is not reachable, we could trivially change the training algorithm to incorporate learning on configurations that result from incorrect transitions, and thereby mitigate the effects of error propagation at parsing time. Conceptually, all that we need to change is line 9. Instead of following the prediction t_p only when it is correct (line 11), we could sometimes choose to follow t_p also when it is not correct.

Algorithm 3 Online training with exploration for greedy transition-based parsers (i th iteration)

```
1: for sentence  $W$  with gold tree  $T$  in corpus do
2:    $c \leftarrow \text{INITIAL}(W)$ 
3:   while not  $\text{TERMINAL}(c)$  do
4:      $\text{CORRECT}(c) \leftarrow \{t \mid o(t; c, T) = \text{true}\}$ 
5:      $t_p \leftarrow \arg \max_{t \in \text{LEGAL}(c)} \mathbf{w} \cdot \phi(c, t)$ 
6:      $t_o \leftarrow \arg \max_{t \in \text{CORRECT}(c)} \mathbf{w} \cdot \phi(c, t)$ 
7:     if  $t_p \notin \text{CORRECT}(c)$  then
8:        $\text{UPDATE}(\mathbf{w}, \phi(c, t_o), \phi(c, t_p))$ 
9:        $c \leftarrow \text{EXPLORE}_{k,p}(c, t_o, t_p, i)$ 
10:    else
11:       $c \leftarrow t_p(c)$ 
1: function  $\text{EXPLORE}_{k,p}(c, t_o, t_p, i)$ 
2:   if  $i > k$  and  $\text{RAND}() < p$  then
3:     return  $t_p(c)$ 
4:   else
5:     return  $\text{NEXT}(c, t_o)$ 
```

The rest of the training algorithm does not need to change, as the set $\text{CORRECT}(c)$ obtained in line 4 would now include the set of optimal transitions to take from configurations reached by following the incorrect transition, as provided by the new oracle. Following Goldberg and Nivre (2012), we call this approach *learning with exploration*. The modified training procedure is specified in Algorithm 3.

There are three major questions that need to be answered when implementing a concrete version of this algorithm:

Exploration Policy When do we follow an incorrect transition, and which one do we follow?

Optimality What constitutes an *optimal* transition in configurations from which the gold tree is not reachable?

Oracle Given a definition of optimality, how do we calculate the set of optimal transitions in a given configuration?

The first two questions are independent of the specific transition system. In our experiments, we use a simple exploration policy, parameterized by an iteration number k and a probability p . This policy always chooses an oracle transition during the first k iterations but later chooses the oracle transition with

probability $1 - p$ and the (possibly incorrect) model prediction otherwise. This is defined in the function $\text{EXPLORE}_{k,p}(c, t_o, t_p, i)$ (called in line 9 of Algorithm 3), which takes two additional arguments compared to Algorithm 2: the model prediction t_p and the current training iteration i . If i exceeds the iteration threshold k and if a randomly generated probability does not exceed the probability threshold p , then the function returns $t_p(c)$, which means that we follow the (incorrect) model prediction. Otherwise, it reverts to the old $\text{NEXT}(c, t_o)$ function, returning c for easy-first and $t_o(c)$ for the other systems. We show in Section 5 that the training procedure is relatively insensitive to the choice of k and p values as long as predicted transitions are chosen often.

Our optimality criterion is directly related to the attachment score metrics commonly used to evaluate dependency parsers.⁴ We say that a transition t is optimal in a configuration c if and only if the best achievable attachment score from $t(c)$ is equal to the best achievable attachment score from c .

The implementation of oracles is specific to each transition system. In the next section, we first provide a characterization of *complete non-deterministic oracles*, also called *dynamic oracles*, which is what we require for the training procedure in Algorithm 3. We then define a property of transition systems which we call *arc decomposition* and present a general method for deriving complete non-deterministic oracles for arc-decomposable systems. Finally, we use this method to derive concrete oracles for the arc-eager, hybrid and easy-first systems, which are all arc-decomposable. In Section 5, we then show experimentally that we indeed achieve better parsing accuracy when using exploration during training.

4 Oracles for Transition-Based Parsing

Almost all greedy transition-based parsers described in the literature are trained using what we call *static* oracles. We now make this notion precise and contrast it with *non-deterministic* and *complete* oracles. Following the terminology of Goldberg and Nivre

⁴The labeled attachment score (LAS) is the percentage of words in a sentence that are assigned both the correct head and the correct label. The unlabeled attachment score (UAS) is the percentage of words that are assigned the correct head (regardless of label).

(2012), we reserve the term *dynamic oracles* for oracles that are both non-deterministic and complete.

4.1 Characterizing Oracles

During training, we assume that the oracle is a boolean function $o(t; c, T)$, which returns **true** if and only if transition t is correct in configuration c for gold tree T (cf. Algorithms 2–3). However, such a function may be defined in terms of different underlying functions that we also call oracles.

A *static* oracle is a function $o_s(T)$ mapping a tree T to a sequence of transitions t_1, \dots, t_n . A static oracle is correct if starting in the initial configuration and applying the transitions in $o_s(T)$ in order results in the transition system reaching a terminating configuration with parse tree T . Formally, a static oracle is correct if and only if, for every projective dependency tree T with yield W , $o_s(T) = t_1, \dots, t_n$, $c = t_n(\dots(t_1(\text{INITIAL}(W))))$, $\text{TERMINAL}(c)$ and $A_c = T$.⁵ When using a static oracle for training in Algorithm 2, the function $o(t; c, T)$ returns **true** if $o_s(T) = t_1, \dots, t_n$, $c = t_{i-1}(\dots(t_1(\text{INITIAL}(W))))$ (for some i , $1 \leq i \leq n$) and $t = t_i$. If $t \neq t_i$, $o(t; c, T) = \mathbf{false}$; if $c \neq t_{i-1}(\dots(t_1(\text{INITIAL}(W))))$ (for all i , $1 \leq i \leq n$), $o(t; c, T)$ is undefined. A static oracle is therefore essentially incomplete, because it is only defined for configurations that are part of the oracle path.⁶ Static oracles either allow a single transition at a given configuration, or are undefined for that configuration.

By contrast, a *non-deterministic* oracle is a function $o_n(c, T)$ mapping a configuration c and a tree T to a *set* of transitions. A non-deterministic oracle is correct if and only if, for every projective dependency tree T , every configuration c from which T is reachable, and every transition $t \in o_n(c, T)$, $t(c)$ is a configuration from which T is still reachable. Note that this definition of correctness for non-deterministic oracles is restricted to configurations from which a goal tree is reachable. Non-

⁵Since all the transition systems considered in this paper are restricted to projective dependency trees, we only define correctness with respect to this class. There are obvious generalizations that apply to more expressive transition systems.

⁶Static oracles are usually described as rules over parser configurations, i.e., “if the configuration is X take transition Y”, giving the impressions they are functions from configurations to transitions. However, as explained here, these rules are only correct if the sequence of transitions is followed in its entirety.

deterministic oracles are more flexible than static oracles in that they allow for spurious ambiguity: they support the possibility of different sequences of transitions leading to the gold tree. However, they are still only guaranteed to be correct on a subset of the possible configurations. Thus, when using a non-deterministic oracle for training in Algorithm 2, the function $o(t; c, T)$ returns **true** if T is reachable from c and $t \in o_n(c, T)$. However, if T is not reachable from c , $o(t; c, T)$ is not necessarily well-defined.

A *complete* non-deterministic oracle is a function $o_d(c, T)$ for which this restriction is removed, so that correctness is defined over all configurations that are reachable from the initial configuration. Following Goldberg and Nivre (2012), we call complete non-deterministic oracles *dynamic*. In order to define correctness for dynamic oracles, we must first introduce a *cost function* $\mathcal{C}(A, T)$, which measures the cost of outputting parse A when the gold tree is T . In this paper, we define cost as Hamming loss (for labeled or unlabeled dependency arcs), which is directly related to the attachment score metrics used to evaluate dependency parsers, but other cost functions are conceivable. We say that a complete non-deterministic oracle is correct if and only if, for every projective dependency tree T with yield W , every configuration c that is reachable from $\text{INITIAL}(W)$, and every transition $t \in o_d(c, T)$, $\min_{A: c \rightsquigarrow A} \mathcal{C}(A, T) = \min_{A: t(c) \rightsquigarrow A} \mathcal{C}(A, T)$, where $c \rightsquigarrow A$ signifies that the parse A is reachable from c , a notion that will be formally defined in the next subsection. In other words, even if the gold tree T is no longer reachable itself, the best tree reachable from $t(c)$ has the same cost as the best tree reachable from c .

In addition to a cost function for arc sets and trees, it is convenient to define a cost function for transitions. We define $\mathcal{C}(t; c, T)$ to be the difference in cost between the best tree reachable from $t(c)$ and c , respectively. That is:

$$\mathcal{C}(t; c, T) = \min_{A: t(c) \rightsquigarrow A} \mathcal{C}(A, T) - \min_{A: c \rightsquigarrow A} \mathcal{C}(A, T)$$

A dynamic oracle can then be defined as an oracle that returns the set of transitions with zero cost:

$$o_d(c, T) = \{t \mid \mathcal{C}(t; c, T) = 0\}$$

4.2 Arc Reachability and Arc Decomposition

We now define the notion of reachability for parses (or arc sets), used already in the previous subsection, and relate it to reachability for individual dependency arcs. This enables us to define a property of transition systems called arc decomposition, which is very useful when deriving dynamic oracles.

Arc Reachability We say that a dependency arc (h, d) ⁷ is *reachable* from a configuration c , written $c \rightsquigarrow (h, d)$, if there is a (possibly empty) sequence of transitions t_1, \dots, t_k such that $(h, d) \in A_{(t_k(\dots t_1(c)))}$. In words, we require a sequence of transitions starting from c and leading to a configuration whose arc set contains (h, d) .

Arc Set Reachability A set of dependency arcs $A = \{(h_1, d_1), \dots, (h_n, d_n)\}$ is *reachable* from a configuration c , written $c \rightsquigarrow A$, if there is a (possibly empty) sequence of transitions t_1, \dots, t_k such that $A \subseteq A_{(t_k(\dots t_1(c)))}$. In words, there is a sequence of transitions starting from c and leading to a configuration where all arcs in A have been derived.

Tree Consistency A set of arcs A is said to be *tree consistent* if there exists a projective dependency tree T such that $A \subseteq T$.

Arc Decomposition A transition system is said to be *arc decomposable* if, for every tree consistent arc set A and configuration c , $c \rightsquigarrow A$ is entailed by $c \rightsquigarrow (h, d)$ for every arc $(h, d) \in A$. In words, if every arc in a tree consistent arc set is reachable from a configuration, then the entire arc set is also reachable from that configuration.

Arc decomposition is a powerful property, allowing us to reduce reasoning about the reachability of arc sets or trees to reasoning about the reachability of individual arcs, and will later use this property to derive dynamic oracles for the arc-eager, hybrid and easy-first systems.

⁷We consider unlabeled arcs here in order to keep notation simple. Everything is trivially extendable to the labeled case.

4.3 Proving Arc Decomposition

Let us now sketch how arc decomposition can be proven for the transition systems in consideration.

Arc-Eager For the arc-eager system, consider an arbitrary configuration $c = (\sigma, \beta, A)$ and a tree-consistent arc set A' such that all arcs are reachable from c . We can partition A' into four sets, each of which is by necessity itself a tree-consistent arc-set:

- (1) $\bar{B} = \{(h, d) \mid h, d \notin \beta\}$
- (2) $B = \{(h, d) \mid h, d \in \beta\}$
- (3) $B_h = \{(h, d) \mid h \in \beta, d \in \sigma\}$
- (4) $B_d = \{(h, d) \mid d \in \beta, h \in \sigma\}$

Arcs in \bar{B} are already in A and cannot interfere with other arcs. B is reachable by any sequence of transitions that derives a tree consistent with B for a sentence containing only the words in β . In deriving this tree, every node x involved in some arc in B_h or B_d must at least once be at the head of the buffer. Let c_x be the first such configuration. From c_x , every arc $(x, d) \in B_h$ can be derived without interfering with arcs in A' by a sequence of REDUCE and LEFT-ARC_{lb} transitions. This sequence of transitions will trivially not interfere with other arcs in B_h . Moreover, it will not interfere with arcs in B_d because A' is tree consistent and projectivity ensures that an arc of the form (y, z) ($y \in \sigma, z \in \beta$) must satisfy $y < d < x \leq z$. Finally, it will not interfere with arcs in B because the buffer remains unchanged. After deriving every arc $(x, d) \in B_h$, we remain with at most one $(h, x) \in B_d$ (because of the single-head constraint). By the same reasoning as above, a sequence of REDUCE and LEFT-ARC_{lb} transitions will take us to a configuration where h is on top of the stack without interfering with arcs in A' . We can then derive the arc (h, x) using RIGHT-ARC_{lb}. This does not interfere with arcs remaining in B_h or B_d because all such arcs must have their buffer node further down the buffer (due to projectivity). At this point, we have reached a configuration c_{x+1} to which the same reasoning applies for the next node $x + 1$.

Hybrid The proof for the hybrid system is very similar but with a slightly different partitioning because of the bottom-up order and the different way of handling right-arcs.

Easy-First For the easy-first system, we only need to partition arcs into $\bar{\mathcal{L}} = \{(h, d) \mid d \notin \lambda\}$ and $\mathcal{L} = \{(h, d) \mid h, d \in \lambda\}$. The former must already be in A , and for the latter there can be no conflict between arcs as long as we respect the bottom-up ordering.

Arc-Standard Unfortunately, arc decomposition does *not* hold for the arc-standard system. To see why, consider a configuration with the stack $\sigma = a, b, c$. The arc (c, b) is reachable via LEFT, the arc (b, a) is reachable via RIGHT, LEFT, the arc set $A = \{(c, b), (b, a)\}$ forms a projective tree and is thus tree consistent, but it is easy to convince oneself that A is not reachable from this configuration. The reason that the above proof technique fails for the arc-standard system is that the arc set corresponding to \mathcal{B} in the arc-eager system may involve arcs where both nodes are still on the stack, and we cannot guarantee that all projective trees consistent with these arcs can be derived. In the very similar hybrid system, such arcs exist as well but they are limited to arcs of the form (h, d) where $h < d$ and h and d are adjacent on the stack, and this restriction is sufficient to restore arc decomposition.

4.4 Deriving Oracles

We now present a procedure for deriving a dynamic oracle for any arc-decomposable system. First of all, we can define a non-deterministic oracle as follows:

$$o_n(c, T) = \{t \mid t(c) \rightsquigarrow T\}$$

That is, we allow all transitions after which the goal tree is still reachable. Note that if $c \rightsquigarrow T$ holds, then the set returned by the oracle is guaranteed to be non-empty. For a sound and complete transition system, we know that $\text{INITIAL}(W) \rightsquigarrow T$ for any projective dependency tree with yield W , and the oracle is guaranteed to return a non-empty set as long as we are not in the terminal configuration and have followed transitions suggested by the oracle.

In order to extend the non-deterministic oracle to a dynamic oracle, we make use of the transition cost function introduced earlier:

$$o_d(c, T) = \{t \mid \mathcal{C}(t; c, T) = 0\}$$

As already mentioned, we assume here that the cost is the difference in Hamming loss between the best

tree reachable before and after the transition.⁸ Assuming arc decomposition, this is equivalent to the number of gold arcs that are reachable before but not after the transition. For configurations from which T is reachable, the dynamic oracle coincides with the non-deterministic oracle. But for configurations from which T cannot be derived, the dynamic oracle returns transitions leading to the best parse A (in terms of Hamming distance from T) which is reachable from c . This is the behavior expected from a dynamic oracle, as defined in Section 4.1.

Thus, in order to derive a dynamic oracle for an arc-decomposable transition system, it is sufficient to show that the transition cost function $\mathcal{C}(t; c, T)$ can be computed efficiently for that system.⁹ Next we show how to do this for the arc-eager, hybrid and easy-first systems.

4.5 Concrete Oracles

In a given transition system, the set of individually reachable arcs is relatively straightforward to compute. In an arc-decomposable system, we know that any intersection of the set of individually reachable arcs with a projective tree is tree consistent, and therefore also reachable. In particular, this holds for the goal tree. For such systems, we can therefore compute the transition cost by intersecting the set of arcs that are individually reachable from a configuration with the goal arc set, and see how a given transition affects this set of reachable arcs.

Arc-Eager In the arc-eager system, an arc (h, d) is reachable from a configuration c if one of the following conditions hold:

- (1) (h, d) is already derived ($(h, d) \in A_c$);
- (2) h and d are in the buffer;
- (3) h is on the stack and d is in the buffer;
- (4) d is on the stack and is not assigned a head and h is in the buffer.

⁸The framework is easily adapted to a different cost function such as weighted Hamming cost, where different gold arcs are weighted differently.

⁹In fact, in order to use the dynamic oracle with our current learning algorithm, we do not need the full power of the *cost* function: it is sufficient to distinguish between transitions with zero cost and transitions with non-zero cost.

The cost function for a configuration of the form $c = (\sigma|s, b|\beta, A)$ ¹⁰ can be calculated as follows:¹¹

- $\mathcal{C}(\text{LEFT}; c, T)$: Adding the arc (b, s) and popping s from the stack means that s will not be able to acquire any head or dependents in β . The cost is therefore the number of arcs in T of the form (k, s) or (s, k) such that $k \in \beta$. Note that the cost is 0 for the trivial case where $(b, s) \in T$, but also for the case where b is not the gold head of s but the real head is not in β (due to an erroneous previous transition) and there are no gold dependents of s in β .
- $\mathcal{C}(\text{RIGHT}; c, T)$: Adding the arc (s, b) and pushing b onto the stack means that b will not be able to acquire any head in σ or β , nor any dependents in σ . The cost is therefore the number of arcs in T of the form (k, b) , such that $k \in \sigma \cup \beta$, or of the form (b, k) such that $k \in \sigma$ and there is no arc (x, k) in A_c . Note again that the cost is 0 for the trivial case where $(s, b) \in T$, but also for the case where s is not the gold head of b but the real head is not in σ or β (due to an erroneous previous transition) and there are no gold dependents of b in σ .
- $\mathcal{C}(\text{REDUCE}; c, T)$: Popping s from the stack means that s will not be able to acquire any dependents in $B = b|\beta$. The cost is therefore the number of arcs in T of the form (s, k) such that $k \in B$. While it may seem that a gold arc of the form (k, s) should be accounted for as well, note that a gold arc of that form, if it exists, is already accounted for by a previous (erroneous) RIGHT transition when s acquired its head.
- $\mathcal{C}(\text{SHIFT}; c, T)$: Pushing b onto the stack means that b will not be able to acquire any head or dependents in $S = s|\sigma$. The cost is therefore the number of arcs in T of the form (k, b) or (b, k) such that $k \in S$ and (for the second case) there is no arc (x, k) in A_c .

¹⁰This is a slight abuse of notation, since for the SHIFT transition s may not exist, and for the REDUCE transition b may not exist.

¹¹While very similar to the presentation in Goldberg and Nivre (2012), this version includes a small correction to the RIGHT and SHIFT transitions.

Hybrid In the hybrid system, an arc (h, d) is reachable from a configuration c if one of the following conditions holds:

- (1) (h, d) is already derived ($(h, d) \in A_c$);
- (2) h and d are in the buffer;
- (3) h is on the stack and d is in the buffer;
- (4) d is on the stack and h is in the buffer;
- (5) d is in stack location i , h is in stack location $i - 1$ (that is, the stack has the form $\sigma = \dots, h, d, \dots$).

The cost function for a configuration of the form $c = (\sigma|s_1|s_0, b|\beta, A)$ ¹² can be calculated as follows:

- $\mathcal{C}(\text{LEFT}; c, T)$: Adding the arc (b, s_0) and popping s_0 from the stack means that s_0 will not be able to acquire heads from $H = \{s_1\} \cup \beta$ and will not be able to acquire dependents from $D = \{b\} \cup \beta$. The cost is therefore the number of arcs in T of the form (s_0, d) and (h, s_0) for $h \in H$ and $d \in D$.
- $\mathcal{C}(\text{RIGHT}; c, T)$: Adding the arc (s_1, s_0) and popping s_0 from the stack means that s_0 will not be able to acquire heads or dependents from $B = \{b\} \cup \beta$. The cost is therefore the number of arcs in T of the form (s_0, d) and (h, s_0) for $h, d \in B$.
- $\mathcal{C}(\text{SHIFT}; c, T)$: Pushing b onto the stack means that b will not be able to acquire heads from $H = \{s_1\} \cup \sigma$, and will not be able to acquire dependents from $D = \{s_0, s_1\} \cup \sigma$. The cost is therefore the number of arcs in T of the form (b, d) and (h, b) for $h \in H$ and $d \in D$.

Easy-First In the easy-first system, an arc (h, d) is reachable from a configuration c if one of the following conditions holds:

- (1) (h, d) is already derived ($(h, d) \in A_c$);
- (2) h and d are in the list λ .

When adding an arc (h, d) , d is removed from the list λ and cannot participate in any future arcs. Thus, a transition has a cost > 0 with respect to a tree T if one of the following holds:

¹²Note again that s_0 may be missing in the case of SHIFT case and s_1 in the case of SHIFT and LEFT.

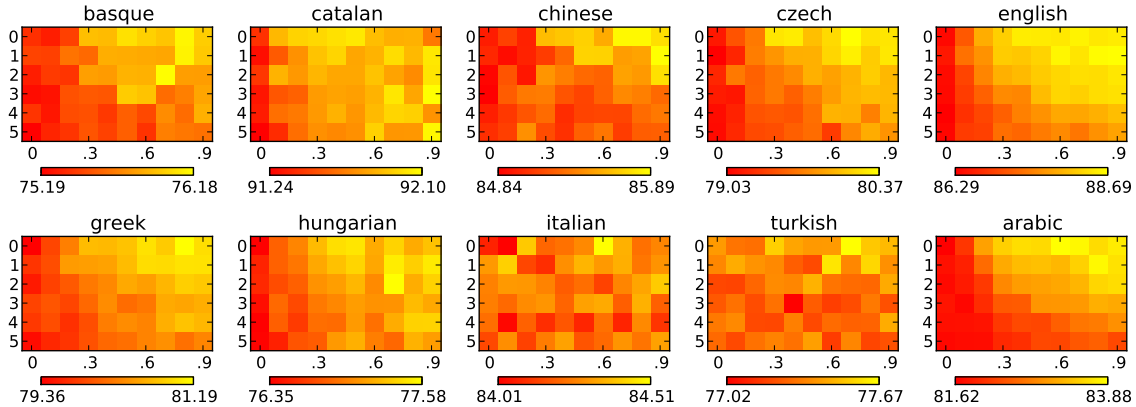


Figure 1: Effect of k (y axis) and p (x axis) values on parsing accuracies for the arc-eager system on the various CoNLL-2007 shared-task languages. Each point is an average UAS of 5 runs with different seeds. The general trend is that smaller k and higher p are better.

- (1) it adds an arc (h, d) such that $(h', d) \in T$ for some $h' \in \lambda$, $h' \neq h$;
- (2) it adds an arc (h, d) such that $(d, d') \in T$ for some $d' \in \lambda$.

The exact cost can be calculated by counting the number of such arcs.

5 Experiments and Results

Setup, data and parameters The goal of our experiments is to evaluate the utility of the dynamic oracles for training, by comparing a training scenario which only sees configurations that can lead to the gold tree (following a static oracle for the left-to-right systems and a non-deterministic but incomplete oracle for the easy-first system), against a training scenario that involves exploration of incorrect states, using the dynamic oracles.

As our training algorithm involves a random component (we shuffle the sentences prior to each iteration, and randomly select whether to follow a correct or incorrect action), we evaluate each setup five times using different random seeds, and report the averaged results.

We perform all of the experiments on the multi-lingual CoNLL-2007 data sets. We use 15 training iterations for the left-to-right parsers, and 20 training iterations for the easy-first parser. We use the standard perceptron update as our update rule in training, and use the averaged weight vector for prediction in test time. The feature sets differ by transition system but are kept the same across data sets. The ex-

act feature-set definitions for the different systems are available in the accompanying software, which is available on line at the first author’s homepage.

Effect of exploration parameters In an initial set of experiments, we investigate the effect of the exploration parameters k and p on the arc-eager system. The results are presented in Figure 1. While the optimal parameters vary by data set, there is a clear trend toward lower values of k and higher values of p . This is consistent with the report of Goldberg and Nivre (2012) who used a fixed small value of k and large value of p throughout their experiments.

Training with exploration for the various systems For the second experiment, in which we compared training with a static oracle to training with exploration, we fixed the exploration parameters to $k = 1$ and $p = 0.9$ for all data sets and transition-system combinations. The results in terms of labeled accuracies (for the left-to-right systems) and unlabeled accuracies (for all systems) are presented in Table 1. Training with exploration using the dynamic oracles yields improved accuracy for the vast majority of the setups. The notable exceptions are the arc-eager and easy-first systems for unlabeled Italian and the arc-hybrid system in Catalan, where we observe a small drop in accuracy. However, we can safely conclude that training with exploration is beneficial and note that we may get even further gains in the future using better methods for tuning the exploration parameters or better training methods.

system / language	hungarian	chinese	greek	czech	basque	catalan	english	turkish	arabic	italian
UAS										
eager:static	76.42	85.01	79.53	78.70	75.14	91.30	86.10	77.38	81.59	84.40
eager:dynamic	77.48	85.89	80.98	80.25	75.97	92.02	88.69	77.39	83.62	84.30
hybrid:static	76.39	84.96	79.40	79.71	73.18	91.30	86.43	75.91	83.43	83.43
hybrid:dynamic	77.54	85.10	80.49	80.07	73.70	91.06	87.62	76.90	84.04	83.83
easyfirst:static	81.27	87.01	81.28	82.00	75.01	92.50	88.57	78.92	82.73	85.31
easyfirst:dynamic	81.52	87.48	82.25	82.39	75.87	92.85	89.41	79.29	83.70	85.11
LAS										
eager:static	66.72	81.24	72.44	71.08	65.34	86.02	84.93	66.59	72.10	80.17
eager:dynamic	68.41	82.23	73.81	72.99	66.63	86.93	87.69	67.05	73.92	80.43
hybrid:static	66.54	80.17	70.99	71.88	62.84	85.57	84.96	64.80	73.16	78.78
hybrid:dynamic	68.05	80.59	72.07	72.15	63.52	85.47	86.28	66.12	74.10	79.25

Table 1: Results on the CoNLL 2007 data set. UAS, including punctuation. Each number is an average over 5 runs with different randomization seeds. All experiments used the same exploration parameters of $k=1$, $p=0.9$.

6 Related Work

The error propagation problem for greedy transition-based parsing was diagnosed by McDonald and Nivre (2007) and has been tackled with a variety of techniques including parser stacking (Nivre and McDonald, 2008; Martins et al., 2008) and beam search and structured prediction (Zhang and Clark, 2008; Zhang and Nivre, 2011). The technique called bootstrapping in Choi and Palmer (2011) is similar in spirit to training with exploration but is applied iteratively in batch mode and is only approximate due to the use of static oracles. Dynamic oracles were first explored by Goldberg and Nivre (2012).

In machine learning more generally, our approach can be seen as a problem-specific instance of *imitation learning* (Abbeel and Ng, 2004; Vlachos, 2012; He et al., 2012; Daumé III et al., 2009; Ross et al., 2011), where the dynamic oracle is used to implement the *optimal expert* needed in the imitation learning setup. Indeed, our training procedure is closely related to DAGger (Ross et al., 2011), which also trains a classifier to match an expert on a distribution of possibly suboptimal states obtained by running the system itself. Our training procedure can be viewed as an online version of DAGger (He et al., 2012) with two extensions: First, our learning algorithm involves a stochastic policy parameterized by k , p for choosing between the oracle or the model prediction, whereas DAGger always follows the system’s own prediction (essentially running with $k = 0$, $p = 1$). The heatmaps in Figure

1 show that this parameterization is beneficial. Second, while DAGger assumes an expert providing a single label at each state, our oracle is nondeterministic and allows multiple correct labels (transitions) which our training procedure tie-breaks according to the model’s current prediction, a technique that has recently been proposed in an extension to DAGger by He et al. (2012). Other related approaches in the machine learning literature include stacked sequential learning (Cohen and Carvalho, 2005), LaSO (Daumé III and Marcu, 2005), Searn (Daumé III et al., 2009) and SMILe (Ross and Bagnell, 2010).

7 Conclusion

In this paper, we have extended the work on dynamic oracles presented in Goldberg and Nivre (2012) in several directions by giving formal characterizations of non-deterministic and complete oracles, defining the arc-decomposition property for transition systems, and using this property to derive novel complete non-deterministic oracles for the hybrid and easy-first systems (as well as a corrected oracle for the arc-eager system). We have then used the completeness of these new oracles to improve the training procedure of greedy parsers to include explorations of configurations which result from incorrect transitions. For all three transition systems, we get substantial accuracy improvements on many languages. As the changes all take place at training time, the very fast running time of the greedy algorithm at test time is maintained.

References

- Pieter Abbeel and Andrew Y Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, page 1.
- Jinho D. Choi and Martha Palmer. 2011. Getting the most out of transition-based dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 687–692.
- William W. Cohen and Vitor R. Carvalho. 2005. Stacked sequential learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 671–676.
- Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 169–176.
- Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning*, 75:297–325.
- Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT)*, pages 742–750.
- Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING)*, pages 959–976.
- He He, Hal Daumé III, and Jason Eisner. 2012. Imitation learning by coaching. In *Advances in Neural Information Processing Systems 25*.
- Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1077–1086.
- Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1–11.
- Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic programming algorithms for transition-based dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 673–682.
- André Filipe Martins, Dipanjan Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking dependency parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 157–166.
- Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 122–131.
- Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 91–98.
- Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 950–958.
- Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160.
- Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together (ACL)*, pages 50–57.
- Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34:513–553.
- Stéphane Ross and J. Andrew Bagnell. 2010. Efficient reductions for imitation learning. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 661–668.
- Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pages 627–635.
- Andreas Vlachos. 2012. An investigation of imitation learning algorithms for structured prediction. In *Proceedings of the European Workshop on Reinforcement Learning (EWRL)*, pages 143–154.
- Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 562–571.
- Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193.

Joint Incremental Disfluency Detection and Dependency Parsing

Matthew Honnibal

Department of Computing
Macquarie University
Sydney, Australia

matthew.honnibal@mq.edu.edu.au

Mark Johnson

Department of Computing
Macquarie University
Sydney, Australia

mark.johnson@mq.edu.edu.au

Abstract

We present an incremental dependency parsing model that jointly performs disfluency detection. The model handles speech repairs using a novel non-monotonic transition system, and includes several novel classes of features. For comparison, we evaluated two pipeline systems, using state-of-the-art disfluency detectors. The joint model performed better on both tasks, with a parse accuracy of 90.5% and 84.0% accuracy at disfluency detection. The model runs in expected linear time, and processes over 550 tokens a second.

1 Introduction

Most unscripted speech contains filled pauses (*ums* and *uhs*), and errors that are usually edited on-the-fly by the speaker. Disfluency detection is the task of detecting these infelicities in spoken language transcripts. The task has some immediate value, as disfluencies have been shown to make speech recognition output much more difficult to read (Jones et al., 2003), but has also been motivated as a module in a natural language understanding pipeline, because disfluencies have proven problematic for PCFG parsing models.

Instead of a pipeline approach, we build on recent work in transition-based dependency parsing, to perform the two tasks jointly. There have been two small studies of dependency parsing on unscripted speech, both using entirely greedy parsing strategies, without a direct comparison against a pipeline architecture (Jorgensen, 2007; Rasooli and Tetreault, 2013). We go substantially beyond these pilot studies, and present a system that compares favourably to a pipeline consisting of state-of-the-art components. Our parser largely follows

the design of Zhang and Clark (2011). We use a structured averaged perceptron model with beam-search decoding (Collins, 2002). Our feature set is based on Zhang and Clark (2011), and our transition-system is based on the arc-eager system of Nivre (2003).

We extend the transition system with a novel non-monotonic transition, Edit. It allows sentences like ‘*Pass the pepper uh salt*’ to be parsed incrementally, without the need to guess early that *pepper* is disfluent. This is achieved by re-processing the leftward children of the word Edit marks as disfluent. For instance, if the parser attaches *the* to *pepper*, but subsequently marks *pepper* as disfluent, *the* will be returned to the stack. We also exploit the ease with which the model can incorporate arbitrary features, and design a set of features that capture the ‘rough copy’ structure of some speech repairs, which motivated the Johnson and Charniak (2004) noisy channel model.

Our main comparison is against two pipeline systems, which use the two current state-of-the-art disfluency detection systems as pre-processors to our parser, minus the custom disfluency features and transition. The joint model compared favourably to the pipeline parsers at both tasks, with an unlabelled attachment score of 90.5%, and 84.0% accuracy at detecting speech repairs. An efficient implementation is available under an open-source license.¹ The future prospects of the system are also quite promising. Because the parser is incremental, it should be well suited to unsegmented text such as the output of a speech-recognition system. We consider our main contributions to be:

- a novel non-monotonic transition system, for speech repairs and restarts,

¹<http://github.com/syllog1sm/redshift>

A flight to um Boston I mean Denver Tuesday
 FP RM IM RP

Figure 1: A sentence with disfluencies annotated in the style of Shriberg (1994) and the Switchboard corpus. FP=Filled Pause, RM=Reparandum, IM=Interregnum, RP=Repair. We follow previous work in evaluating the system on the accuracy with which it identifies speech-repairs, marked *reparandum* above.

- several novel feature classes,
- direct comparison against the two best disfluency pre-processors, and
- state-of-the-art accuracy for both speech parsing and disfluency detection.

2 Switchboard Disfluency Annotations

The Switchboard portion of the Penn Treebank (Marcus et al., 1993) consists of telephone conversations between strangers about an assigned topic. Two annotation layers are provided: one for syntactic bracketing (MRG files), and one for disfluencies (DPS files). The disfluency layer marks elements with little or no syntactic function, such as filled pauses and discourse markers, and annotates speech repairs using the Shriberg (1994) system of reparandum/interregnum/repair. An example is shown in Figure 1.

In the syntactic annotation, edited words are covered by a special node labelled EDITED. The idea is to mark text which, if excised, would result in a grammatical sentence. The MRG files do not mark other types of disfluencies. We follow the evaluation defined by Charniak and Johnson (2001), which evaluates the accuracy of identifying speech repairs and restarts. This definition of the task is the standard in recent work. The reason for this is that filled pauses can be detected using a simple rule-based approach, and parentheticals have less impact on readability and down-stream processing accuracy.

The MRG and DPS layers have high but imperfect agreement over what tokens they mark as speech repairs: of the text annotated with both layers, 33,720 tokens are marked as disfluent in at least one layer, 32,310 are only marked as disfluent by the DPS files, and 32,742 are only marked as disfluent by the MRG layer.

The Switchboard annotation project was not fully completed. Because disfluency annotation is cheaper to produce, many of the DPS training files do not have matching MRG files. Only 619,236 of the 1,482,845 tokens in the DPS disfluency-

detection training data have gold-standard syntactic parses. Our system requires the more expensive syntactic annotation, but we find that it outperforms the previous state-of-the-art (Qian and Liu, 2013), despite training on less than half the data.

2.1 Dependency Conversion

As is standard in statistical dependency parsing of English, we acquire our gold-standard dependencies from phrase-structure trees. We used the 2013-04-05 version of the Stanford dependency converter (de Marneffe et al., 2006). As is standard for English dependency parsing, we use the Basic Dependencies scheme, which produces strictly projective representations.

At first we feared that the filled pauses, disfluencies and meta-data tokens in the Switchboard corpus might disrupt the conversion process, by making it more difficult for the converter to recognise the underlying production rules.

To test this, we performed a small experiment. We prepared two versions of the corpus: one where EDITED nodes, filled pauses and meta-data were removed before the trees were transformed by the Stanford converter, and one where the disfluency removal was performed after the dependency conversion. The resulting corpora were largely identical: 99.54% of unlabelled and 98.7% of labelled dependencies were the same. The fact that the Stanford converter is quite robust to disfluencies was useful for our baseline joint model, which is trained on dependency trees that also included governors for disfluent words.

We follow previous work on disfluency detection by lower-casing the text and removing punctuation and partial words (words tagged XX and words ending in '-'). We also remove one-token sentences, as their syntactic analyses are trivial. We found that two additional simple pre-processes improved our results: discarding all ‘um’ and ‘uh’ tokens; and merging ‘you know’ and ‘i mean’ into single tokens.

These pre-processes can be completed on the input string without losing information: none of the ‘um’ or ‘uh’ tokens are semantically significant, and the bigrams *you know* and *i mean* have a dependency between the two tokens over 99.9% of the times they occur in the treebank, with *you* and *I* never having any children. This makes it easy to unmerge the tokens deterministically after pars-

ing: all incoming and outgoing arcs will point to *know* or *mean*. The same pre-processing was performed for all our parsing systems.

3 Transition-based Dependency Parsing

A transition-based parser predicts the syntactic structure of a sentence incrementally, by making a sequence of classification decisions. We follow the architecture of Zhang and Clark (2011), who use beam-search for decoding, and a structured averaged perceptron for training. Despite its simplicity, this type of parser has produced highly competitive results on the Wall Street Journal: with the extended feature set described by Zhang and Nivre (2011), it achieves 93.5% unlabelled accuracy on Stanford basic dependencies (de Marneffe et al., 2006). Converting the constituency trees produced by the Charniak and Johnson (2005) reranking parser results in similar accuracy.

Briefly, the transition-based parser consists of a configuration (or ‘state’) which is sequentially manipulated by a set of possible transitions. For us, a state is a 4-tuple $c = (\sigma, \beta, A, D)$, where σ and β are disjoint sets of word indices termed the *stack* and *buffer* respectively, A is the set of dependency arcs, and D is the set of word indices marked disfluent. There are no arcs to or from members of D , so the dependencies and disfluencies can be implemented as a single vector (in our parser, a token is marked as disfluent by setting it as its own head).

We use the arc-eager transition system (Nivre, 2003, 2008), which consists of four parsing actions: **Shift**, **Left-Arc**, **Right-Arc** and **Reduce**. We denote the stack with its topmost element to the right, and the buffer with its first element to the left. A vertical bar is used to indicate concatenation to the stack or buffer, e.g. $\sigma|i$ indicates a stack with the topmost element i and remaining elements σ . A dependency from a governor i to a child j is denoted $i \rightarrow j$. The four arc-eager transitions are shown in Figure 2.

The Shift action moves the first item of the buffer onto the stack. The Right-Arc does the same, but also adds an arc, so that the top two items on the stack are connected. The Reduce move and the Left-Arc both pop the stack, but the Left-Arc first adds an arc from the first word of the buffer to the word on top of the stack. Constraints on the Reduce and Left-Arc moves ensure that every word is assigned exactly one head in the final configuration. We follow the suggestion

$(\sigma, i \beta, A, D) \vdash (\sigma i, \beta, A, D)$	S
$(\sigma i, j \beta, A, D) \vdash (\sigma, j \beta, A \cup \{j \rightarrow i\}, D)$	L
$(\sigma i, j \beta, A, D) \vdash (\sigma i, j, \beta, A \cup \{i \rightarrow j\}, D)$	R
$(\sigma i, \beta, A, D) \vdash (\sigma, \beta, A, D)$	D
Only if i has an incoming arc.	
$(\sigma i, j \beta, A, D) \vdash (\sigma[x_1, x_n], j \beta, A', D')$	E
Where	
$A' = A \setminus \{x \rightarrow y \text{ or } y \rightarrow x : \forall x \in [i, j], \forall y \in \mathbb{N}\}$	
$D' = D \cup [i, j]$	
$x_1 \dots x_n$ are the former left children of i	

Figure 2: Our parser’s transition system. The first four transitions are the standard arc-eager system; the fifth is our novel Edit transition.

of Ballesteros and Nivre (2013) and add a dummy token that governs root dependencies to the end of the sentence. Parsing terminates when this token is at the start of the buffer, and the stack is empty. Disfluencies are added to D via the Edit transition, E, which we now define.

4 A Non-Monotonic Edit Transition

One of the reasons disfluent sentences are hard to parse is that there often appear to be syntactic relationships between words in the reparandum and the fluent sentence. When these relations are considered in addition to the dependencies between fluent words, the resulting structure is not necessarily a projective tree.

Figure 3 shows a simple example, where the repair *square* replaces the reparandum *rectangle*. An incremental parser could easily become ‘garden-pathed’ and attach the repair *square* to the preceding words, constructing the dependencies shown dotted in Figure 3. Rather than attempting to devise an incremental model that avoids constructing such dependencies, we allow the parser to construct these dependencies and later delete them if the governor or child are marked disfluent.

Psycholinguistic models of human sentence processing have long posited *repair* mechanisms (Frazier and Rayner, 1982). Recently, Honnibal et al. (2013) showed that a limited amount of ‘non-monotonic’ behaviour can improve an incremental parser’s accuracy. We here introduce a non-monotonic transition, Edit, for speech repairs.

The Edit transition marks the word i on top of the stack $\sigma|i$ as disfluent, along with its rightward descendents — i.e., all words in the sequence $i \dots j - 1$, where j is the word at the start of the buffer. It then restores the words both preceding and formerly governed by i to the stack.

In other words, the word on top of the stack and

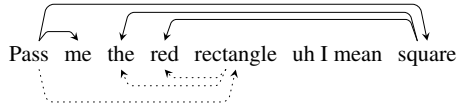


Figure 3: Example where apparent dependencies between the reparandum and the fluent sentence complicate parsing. The dotted edges are difficult for an incremental parser to avoid, but cannot be part of the final parse if it is to be a projective tree. Our solution is to make the transition system non-monotonic: the parser is able to delete edges.

its *rightward descendents* are all marked as disfluent, and the stack is popped. We then restore its leftward children to the stack, and all dependencies to and from words marked disfluent are deleted. The transition is *non-monotonic* in the sense that it can delete dependencies created by a previous transition, and replace tokens onto the stack that had been popped.

Why revisit the leftward children, but not the right? We are concerned about dependencies which might be mirrored between the reparandum and the repair. The rightward subtree of the disfluency might well be incorrect, but if it is, it would still be incorrect if the word on top of the stack were actually fluent. We therefore regard these as parsing errors that we will train our model to avoid. In contrast, avoiding the Left-Arc transitions would require the parser to predict that the head is disfluent when it has not necessarily seen any evidence indicating that.

4.1 Worked Example

Figure 4 shows a gold-standard derivation for a disfluent sentence from the development data. Line 1 shows the state resulting from the initial Shift action. In the next three states, *His* is Left-Arced to *company*, which is then Shifted onto the stack, and Left-Arced to *went* in Line 4.

The dependency between *went* and *company* is not part of the gold-standard, because *went* is disfluent. The correct governor of *company* is the second *went* in the sentence. The Left-Arc move in Line 4 can still be considered correct, however, because the gold-standard analysis is still derivable from the resulting configuration, via the Edit transition. Another non-gold dependency is created in Line 6, between *broke* and *went*, before *broke* is Reduced from the stack in Line 7.

Lines 9 and 10 show the states before and after the Edit transition. The word on top of the stack in Line 9, *went*, has one leftward child, and one right-

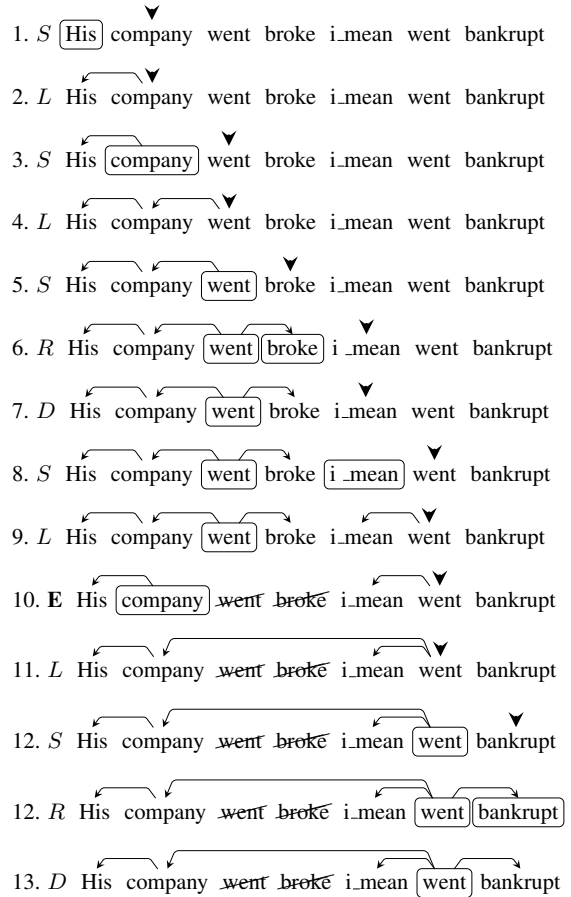


Figure 4: A gold-standard transition sequence using our EDIT transition. Each line specifies an action and shows the state resulting from it. Words on the stack are circled, and the arrow indicates the start of the buffer. Disfluent words are struck-through.

ward child. After the Edit transition is applied, *went* and its rightward child *broke* are both marked disfluent, and *company* is returned to the stack. All of the previous dependencies to and from *went* and *broke* are deleted.

Parsing then proceeds as normal, with the correct governor of *company* being assigned by the Left-Arc in Line 11, and *bankrupt* being Right-Arced to *went* in Line 12. To conserve space, we have omitted the dummy ROOT token, which is placed at the end of the sentence, following the suggestion of Ballesteros and Nivre (2013). The final action will be a Left-Arc from the ROOT token to *went*.

4.2 Dynamic Oracle Training Algorithm

Our non-monotonic transition system introduces substantial *spurious ambiguity*: the gold-standard parse can be derived via many different transition

sequences. Recent work has shown that this can be advantageous (Sartorio et al., 2013; Honnibal et al., 2013; Goldberg and Nivre, 2012), because difficult decisions can sometimes be delayed until more information is available.

Line 5 of Figure 4 shows a state that introduces spurious ambiguity. From this configuration, there are multiple actions that could be considered ‘correct’, in the sense that the gold-standard analysis can be derived from them. The Edit transition is correct because *went* is disfluent, but the Left-Arc and even the Right-Arc are also correct, in that there are continuations from them that lead to the gold-standard analysis.

We regard all transition sequences that can result in the correct analysis as equally valid, and want to avoid stipulating one of them during training. We achieve this by following Goldberg and Nivre (2012) in using a *dynamic oracle* to create partially labelled training data.² A dynamic oracle is a function that determines the cost of applying an action to a state, in terms of gold-standard arcs that are newly unreachable.

We follow Collins (2002) in training an averaged perceptron model to predict transition *sequences*, rather than individual transitions. This type of model is often referred to as a structured perceptron, or sometimes a global perceptron. During training, if the model does not predict the correct sequence, an update is performed, based on the gold-standard sequence and part of the sequence predicted by the current weights. Only part of the sequence is used to calculate the weight update, in order to account for search errors. We use the maximum violation strategy described by Huang et al. (2012) to select the subsequence to update from.

To train our model using the dynamic oracle, we use the latent-variable structured perceptron algorithm described by Sun et al. (2009). Beam-search is performed to find the highest-scoring gold-standard sequence, as well as the highest-scoring prediction. We use the same beam-width for both search procedures.

4.3 Path Length Normalisation

One problem introduced by the Edit transition is that the number of actions applied to a sentence is

² The training data is partially labelled in the sense that instances can have multiple true labels. Equivalently, one might say that the transitions are latent variables, which generate the dependencies.

no longer constant — it is no longer guaranteed to be $2n - 1$, for a sentence of length n . When the Edit transition is applied to a word with leftward children, those children are returned to the stack, and processed again. This has little to no impact on the algorithm’s empirical efficiency, although worst-case complexity is no longer linear, but it does pose a problem for decoding.

The perceptron model tends to assign large positive scores to its top prediction. We thus observed a problem when comparing paths of different lengths, at the end of the sentence. Paths that included Edit transitions were longer, so the sum of their scores tended to be higher.

The same problem has been observed during incremental PCFG parsing, by Zhu et al. (2013). They introduce an additional transition, IDLE, to ensure that paths are the same length. So long as one candidate in the beam is still being processed, all other candidates apply the IDLE transition.

We adopt a simpler solution. We normalise the figure-of-merit for a candidate state, which is used to rank it in the beam, by the length of its transition history. The new figure-of-merit is the arithmetic mean of the candidate’s transition scores, where previously the figure-of-merit was the sum of the candidate’s transition scores.

Interestingly, Zhu et al. (2013) report that they tried exactly this, and that it was less effective than their solution. We found that the features associated with the IDLE transition were uninformative (the state is at termination, so the stack and buffer are empty), and had nothing to do with how many edit transitions were earlier applied.

5 Features for the Joint Parser

Our baseline parser uses the feature set described by Zhang and Nivre (2011). The feature set contains 73 templates that mostly refer to the properties of 12 *context tokens*: the top of the stack (S_0), its two leftmost and rightmost children (S_{0L} , S_{0L2} , S_{0R} , S_{0R2}), its parent and grand-parent (S_{0h} , S_{0h2}), the first word of the buffer and its two leftmost children (N_0 , N_{0L} , N_{0LL}), and the next two words of the buffer (N_1 , N_2).

Atomic features consist of the word, part-of-speech tag, or dependency label for these tokens; and multiple feature atoms are often combined for feature templates. There are also features for the string-distance between S_0 and N_0 , and the left and right valencies (total number of children) of

S0 and N0, as well as the set of their children’s dependency labels. We restrict these to the first and last 2 children for implementation efficiency, as we found this had no effect on accuracy. Numeric features (for distance and valency) are binned with the function $\lambda x : \min(x, 5)$. There is only one bilexical feature template, which pairs the words of S0 and N0. There are also ten tri-tag templates.

Our feature set includes additional dependency label features not used by Zhang and Nivre (2011), as we found that disfluency detection errors often resulted in ungrammatical dependency label combinations. The additional templates combine the POS tag of S0 with two or three dependency labels from its left and right subtrees. Details can be found in the supplementary material.

5.1 Brown Cluster Features

The Brown clustering algorithm (Brown et al., 1992) is a well known source of semi-supervised features. The clustering algorithm is run over a large sample of unlabelled data, to generate a type-to-cluster map. This mapping is then used to generate features that sometimes generalise better than lexical features, and are helpful for out-of-vocabulary words (Turian et al., 2010).

Koo and Collins (2010) found that Brown cluster features greatly improved the performance of a graph-based dependency parser. On our transition-based parser, Brown cluster features bring a small but statistically significant improvement on the WSJ task (0.1-0.3% UAS). Other developers of transition-based parsers seem to have found similar results (personal communication). Since a Brown cluster mapping computed by Liang (2005) is easily available,³ the features are simple to implement and cheap to compute.

Our templates follow Koo and Collins (2010) in including features that refer to cluster prefix strings, as well as the full clusters. We adapt their templates to transition-based parsing by replacing ‘head’ with ‘item on top of the stack’ and ‘child’ with ‘first word of the buffer’. The exact templates can be found in the supplementary material.

The Brown cluster features are used in our ‘baseline’ parser, and in the parsers we use as part of our pipeline systems. They improved development set accuracy by 0.4%. We experimented with the other feature sets in these parsers, but found that they did not improve accuracy on fluent text.

³<http://www.metaoptimize.com/projects/wordreps>

5.2 Rough Copy Features

Johnson and Charniak (2004) point out that in speech repairs, the repair is often a ‘rough copy’ of the reparandum. The simplest case of this is where the repair is a single word repetition. It is common for the repair to differ from the reparandum by insertion, deletion or substitution of one or more words.

To capture this regularity, we first extend the feature-set with three new context tokens:⁴

1. $S0_{re}$: The rightmost edge of S0 descendants;
2. $S0_{le}$: The leftmost edge of S0 descendants;
3. $N0_{le}$: The leftmost edge of N0 descendants.

If a word has no leftward children, it will be its own left-edge, and similarly it will be its own rightward edge if it has no rightward children. Note that the token $S0_{re}$ is necessarily immediately before $N0_{le}$, unless some of the tokens between them are disfluent. We use the $S0_{le}$ and $N0_{le}$ to compute the following rough-copy features:

1. How long is the *prefix word match* between $S0_{le}...S0$ and $N0_{le}...N0$?

If the parser were analysing *the red the blue square*, with *red* on the stack and *square* at N0, its value would be 1.

2. How long is the *prefix POS tag match* between $S0_{le}...S0$ and $N0_{le}...N0$?
3. Do the words in $S0_{le}...S0$ and $N0_{le}...N0$ match exactly?
4. Do the POS tags in $S0_{le}...S0$ and $N0_{le}...N0$ match exactly?

If the parser were analysing *the red square the blue rectangle*, with *square* on the stack and *rectangle* at N0, its value would be *true*.

The prefix-length features are binned using the function $\lambda x : \min(x, 5)$.

5.3 Match Features

This class of features ask which pairs of the *context tokens* match, in word or POS tag. The context tokens in the Zhang and Nivre (2011) feature set are the top of the stack (S0), its head and

⁴As is common in this type of parser, our implementation has a number of vectors for properties that are defined before parsing, such as word forms, POS tags, Brown clusters, etc. A context token is an index into these vectors, allowing features considering these properties to be computed.

grandparent ($S0_h, S0_{h2}$), its two left- and right-most children ($S0_L, S0_{L2}, S0_R, S0_{R2}$), the first three words of the buffer ($N0, N1, N2$), and the two leftmost children of $N0$ ($N0_L, N0_{LL}$). We extend this set with the $S0_{le}, S0_{re}$ and $N0_{le}$ tokens described above, and also the first left and right child of $S0$ and $N0$ ($S0_{L0}, S0_{R0}, N0_{L0}$).

All up, there are 18 context tokens, so $\binom{18}{2} = 153$ token pairs. For each pair of these tokens, we add two binary features, indicating whether the two tokens match in word form or POS tag. We also have two further classes of features: if the words do match, a feature is added indicating the word form; if the tags match, a feature is added indicating the tag. These finer grained versions help the model adjust for the fact that some words can be duplicated in grammatical sentences (e.g. ‘that that’), while most rare words cannot.

5.4 Edited Neighbour Features

Disfluencies are usually string contiguous, even if they do not form a single constituent. In these situations, our model has to make multiple transitions to mark a single disfluency. For instance, if an utterance begins *and the and a*, the stack will contain two entries, for *and* and *the*, and two Edit transitions will be required.

To mitigate this disadvantage of our model, we add four binary features. Two fire when the word or pair of words immediately preceding $N0$ have been marked disfluent; the other two fire when the word or pair of words immediately following $S0$ have been marked disfluent. These features provide an additional string-based view that the parser would otherwise be missing. Speakers tend to be disfluent in bursts: if the previous word is disfluent, the next word is more likely to be disfluent. These four features are therefore all associated with positive weights for the Edit transition. Without these features, we would miss an aspect of disfluency processing that sequence models naturally capture.

6 Part-of-Speech Tagging

We adopt the standard strategy of using a POS tagger as a pre-process before parsing. Most transition-based parsers use a structured averaged perceptron model with beam-search for tagging, as this model achieves competitive accuracy and matches the standard dependency parsing architecture. Our tagger also uses this architecture.

We performed some additional feature engineering for the tagger, in order to improve its accuracy given the lack of case distinctions and punctuation in the data. Our additional features use two sources of unsupervised information. First, we follow the suggestion of Manning (2011) by using Brown cluster features to improve the tagger’s accuracy on unknown words. Second, we compensate for the lack of case distinctions by including features that ask what percentage of the time a word form was seen title-cased, upper-cased and lower-cased in the Google Web1T corpus.

Where most previous work uses cross-fold training for the tagger, to ensure that the parser is trained on tags that reflect run-time accuracies, we do online training of the tagger alongside the parser, using the current tagger model to produce tags during parser training. This had no impact on parse accuracy, and made it slightly easier to develop our tagger alongside the parser.

The tagger achieved 96.5% accuracy on the development data, but when we ran our final test experiments, we found its accuracy dropped to 96.0%, indicating some over-fitting during our feature engineering. On the development data, our parser accuracy improves by about 1% when gold-standard tags are used.

7 Experiments

We use the Switchboard portion of the Penn Treebank (Marcus et al., 1993), as described in Section 2, to train our joint models and evaluate them on dependency parsing and disfluency detection. The pre-processing and dependency conversion are described in Section 2.1. We use the standard train/dev/test split from Charniak and Johnson (2001): Sections 2 and 3 for training, and Section 4 divided into three held-out sections, the first of which is used for final evaluation.

Our parser evaluation uses the SPARSEVAL (Roark et al., 2006) metric. However, we wanted to use the Stanford dependency converter, for the reasons described in Section 2.1, so we used our own implementation. Because we do not need to deal with recognition errors, we do not need to report our parsing results using *P/R/F*-measures. Instead, we report an unlabelled accuracy score, which refers to the percentage of fluent words whose governors were assigned correctly. Note that words marked as disfluent cannot have any incoming or out-going dependencies, so if a word is

incorrectly marked as disfluent, all of its dependencies will be incorrect.

We follow Johnson and Charniak (2004) and others in restricting our disfluency evaluation to speech repairs, which we identify as words that have a node labelled `EDITED` as an ancestor. Unlike most other disfluency detection research, we train only on the MRG files, giving us 619,236 words of training data instead of the 1,482,845 used by the pipeline systems. It may be possible to improve our system’s disfluency detection by leveraging the additional data that does not have syntactic annotation in some way.

All parsing models were trained for 15 iterations. We found that optimising the number of iterations on a development set led to small improvements that did not transfer to a second development set (part of Section 4, which Charniak and Johnson (2001) reserved for ‘future use’).

We test for statistical significance in our results by training 20 models for each experimental configuration, using different random seeds. The random seeds control how the sentences are shuffled during training, which the perceptron model is quite sensitive to. We use the Wilcoxon rank-sums non-parametric test. The standard deviation in UAS for a sample was typically around 0.05%, and 0.5% for disfluency F -measure.

All of our models use beam-search decoding, with a beam width of 32. We found that a beam width of 64 brought a very small accuracy improvement (about 0.1%), at the cost of 50% slower run-time. Wider beams than this brought no accuracy improvement. Accuracy seems to plateau with slightly narrower beams than on newswire text. This is probably due to the shorter sentences in Switchboard.

The baseline and pipeline systems are configured in the same way, except that the baseline parser is modified slightly to allow it to predict disfluencies, using a special dependency label, `ERASED`. All descendants of a word attached to its head by this label are marked as disfluent. Both the baseline and pipeline/oracle parsers use the same feature set: the Zhang and Nivre (2011) features, plus our Brown cluster features.

The baseline system is a standard arc-eager transition-based parser with a structured averaged perceptron model and beam-search decoding. The model is trained in the standard way, with a ‘static’ oracle and maximum-violation update, following

(Huang et al., 2012).

7.1 Comparison with Pipeline Approaches

The accuracy of incremental dependency parsers is well established on the Wall Street Journal, but there are no dependency parsing results in the literature that make it easy to put our joint model’s parsing accuracy into context. We therefore compare our joint model to two pipeline systems, which consist of a disfluency detector, followed by our dependency parser. We also evaluate parse accuracies after oracle pre-processing, to gauge the net effect of disfluencies on our parser’s accuracy.

The dependency parser for the pipeline systems was trained on text with all disfluencies removed, following Charniak and Johnson (2001). The two disfluency detection systems we used were the Qian and Liu (2013) sequence-tagging model, and a version of the Johnson and Charniak (2004) noisy channel model, using the Charniak (2001) syntactic language model and the reranking features of Zwarts and Johnson (2011). They are the two best published disfluency detection systems.

8 Results

Table 1 shows the development set accuracies for our joint parser. Both the disfluency features and the Edit transition make statistically significant improvements, in both disfluency F -measure, unlabelled attachment score (UAS), and labelled attachment score (LAS).

The **Oracle pipeline** system, which uses the gold-standard to clean disfluencies prior to parsing, shows the total impact of speech-errors on the parser. The baseline parser, which uses the Zhang and Nivre (2011) feature set plus the Brown cluster features, scores 1.8% UAS lower than the oracle.

When we add the features described in Sections 5.2, 5.3 and 5.4, the gap is reduced to 1.2% (**+Features**). Finally, the improved transition system reduces the gap further still, to 0.8% UAS (**+Edit transition**). We also tested these features in the Oracle parser, but found they were ineffective on fluent text.

The **w/s** column shows the tokens analysed per second for each system, including disfluencies, with a single thread on a 2.4GHz Intel Xeon. The additional features reduce efficiency, but the non-monotonic Edit transition does not. The system is easily efficient enough for real-time use.

	P	R	F	UAS	LAS	w/s
Baseline joint	79.4	70.1	74.5	89.9	86.9	711
+Features	86.0	77.2	81.3	90.5	87.5	539
+Edit transition	92.2	80.2	85.8	90.9	87.9	555
Oracle pipeline	100	100	100	91.7	88.6	782

Table 1: Development results for the joint models. For the baseline model, disfluencies reduce parse accuracy by 1.7% Unlabelled Attachment Score (UAS). Our features and Edit transition reduce the gap to 0.7%, and improve disfluency detection by 11.3% F -measure.

	Disfl. F	UAS
Johnson et al pipeline	82.1	90.3
Qian and Liu pipeline	83.9	90.1
Baseline joint parser	73.9	89.4
Final joint parser	84.1	90.5

Table 2: Test-set parse and disfluency accuracies. The joint parser is improved by the features and Edit transition, and is better than pre-processing the text with state-of-the-art disfluency detectors.

Table 2 shows the final evaluation. Our main comparison is with the two pipeline systems, described in Section 7.1. The Johnson and Charniak (2004) system was 1.8% less accurate at disfluency detection than the other disfluency detector we evaluated, the state-of-the-art Qian and Liu (2013) system. However, when we evaluated the two systems as pre-processors before our parser, we found that the **Johnson et al pipeline** achieved 0.2% better unlabelled attachment score than the **Qian and Liu pipeline**. We attribute this to the use of the Charniak and Johnson (2001) syntactic language model in the Johnson et al pipeline, which would help the system produce more syntactically consistent output.

Our joint model achieved an unlabelled attachment score of 90.5%, out-performing both pipeline systems. The **Baseline joint parser**, which did not include the Edit transition or disfluency features, scores 1.1% below the **Final joint parser**. All of the parse accuracy differences were found to be statistically significant ($p < 0.001$).

The Edit transition and disfluency features together brought a 10.1% improvement in disfluency F -measure, which was also found to be statistically significant. The final joint parser achieved 0.2% higher disfluency detection accuracy than the previous state-of-the-art, the Qian and Liu (2013) system,⁵ despite having approximately half as much training data (we require syntactic anno-

⁵ Our scores refer to an updated version of the system that corrects minor pre-processing problems. We thank Qian Xian for making his code available.

tation, for which there is less data).

Our significance testing regime involved using 20 different random seeds when training each of our models, which the perceptron algorithm is sensitive to. This could not be applied to the other two disfluency detectors, so we cannot test those differences for significance. However, we note that the 20 samples for our disfluency detector ranged in accuracy from 83.3-84.6, so we doubt that 0.2% mean improvement over the Qian and Liu (2013) result is meaningful.

Although we did not systematically optimise on the development set, our test scores are lower than our development accuracies. Much of the over-fitting seems to be in the POS tagger, which dropped in accuracy by 0.5%.

9 Analysis of Edit Behaviour

In order to understand how the parser applies the Edit transition, we collected some additional statistics over the development data. The parser predicted 2,558 Edit transitions, which together marked 2,706 words disfluent (2,495 correctly). The Edit transition can mark multiple words disfluent when S_0 has one or more rightward descendants. It turns out this case is uncommon; the parser largely assigns disfluency labels word-by-word, only sometimes marking words with rightward descendants as disfluent.

Of the 2,558 Edit transitions, there were 682 cases where at least one leftward child was returned to the stack, and the total number of leftward children returned was 1,132. The most common type of construction that caused the parser to return words to the stack were disfluent predicates, which often have subjects and discourse conjunctions as leftward children. An example of a disfluent predicate with a fluent subject is shown in Figure 4.

There were only 48 cases of the same word being returned to the stack twice. The possibility of words being returned to the stack multiple times is what gives our system worse than linear worst-case complexity. In the worst case, the i th word of a sentence of length n could be returned to the stack $n - (i + 1)$ times. Empirically, the Edit transition made no difference to run-time.

Once a word has been returned to the stack by the Edit transition, how does the parser end up analysing it? If it turned out that almost all of the former leftward children of disfluent words are subsequently marked as disfluent, there would be

little point in returning them to the stack — we could just mark them as disfluent in the original Edit transition. On the other hand, if they are almost all marked as fluent, perhaps they can just be attached as children to the first word of the buffer.

In fact the two cases are almost equally common. Of the 1,132 words returned to the stack, 547 were subsequently marked disfluent, and 584 were not. The parser was also quite accurate in its decisions over these tokens. Of the 547 tokens marked disfluent, 500 were correct — similar to the overall development set precision, 92.2%.

Accuracy over the words returned to the stack might be improved in future by features referring to their former heads. For instance, in *He went broke uh became bankrupt*, we do not currently have features that record the deleted dependency became *he* and *went*. We thank one of the anonymous reviewers for this suggestion.

10 Related Work

The most similar system to ours was published very recently. Rasooli and Tetreault (2013) describe a joint model of dependency parsing and disfluency detection. They introduce a second classification step, where they first decide whether to apply a disfluency transition, or a regular parsing move. Disfluency transitions operate either over a sequence of words before the start of the buffer, or a sequence of words from the start of the buffer forward. Instead of the dynamic oracle training method that we employ, they use a two-stage bootstrap-style process.

Direct comparison between our model and theirs is difficult, as they use the Penn2MALT scheme, and their parser uses greedy decoding, where we use beam search. They also use gold-standard part-of-speech tags, which would improve our scores by around 1%. The use of beam-search may explain much of our performance advantage: they report an unlabelled attachment score of 88.6, and a disfluency detection F -measure of 81.4%. Our training algorithm would be applicable to a beam-search version of their parser, as their transition-system also introduces substantial spurious ambiguity, and some non-monotonic behaviour.

A hybrid transition system would also be possible, as the two types of Edit transition seem to be complementary. The Rasooli and Tetreault system offers a token-based view of disfluencies, which

is useful for examples such as, *and the and the*, which would require two applications of our transition. On the other hand, our Edit transition may have the advantage for more syntactically complicated examples, particularly for disfluent verbs.

The importance of syntactic features for disfluency detection was demonstrated by Johnson and Charniak (2004). Despite this, most subsequent work has used sequence models, rather than syntactic parsers. The other disfluency system that we compare our model to, developed by Qian and Liu (2013), uses a cascade of Maximum Margin Markov Models to perform disfluency detection with minimal syntactic information.

One motivation for sequential approaches is that most applications of these models will be over unsegmented text, as segmenting unpunctuated text is a difficult task that benefits from syntactic features (Zhang et al., 2013).

We consider the most promising aspect of our system to be that it is naturally incremental, so it should be straightforward to extend the system to operate on unsegmented text in subsequent work. Due to its use of syntactic features, from the joint model, the system is substantially more accurate than the previous state-of-the-art in incremental disfluency detection, 77% (Zwarts et al., 2010).

11 Conclusion

We have presented an efficient and accurate joint model of dependency parsing and disfluency detection. The model out-performs pipeline approaches using state-of-the-art disfluency detectors, and is highly efficient, processing over 550 tokens a second. Because the system is incremental, it should be straight-forward to apply it to unsegmented text. The success of an incremental, non-monotonic parser at disfluent speech parsing may also be of some psycholinguistic interest.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments. This research was supported under the Australian Research Council’s Discovery Projects funding scheme (project numbers DP110102506 and DP110102593).

References

Miguel Ballesteros and Joakim Nivre. 2013. Going to the roots of dependency parsing. *Computational Linguistics*. 39:1.

- Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479.
- Eugene Charniak. 2001. Immediate-head parsing for language models. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 124–131. Association for Computational Linguistics, Toulouse, France.
- Eugene Charniak and Mark Johnson. 2001. Edit detection and parsing for transcribed speech. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 118–126. The Association for Computational Linguistics.
- Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n -best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 173–180. Association for Computational Linguistics, Ann Arbor, Michigan.
- Michael 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*, pages 1–8. Association for Computational Linguistics.
- Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*.
- Lyn Frazier and Keith Rayner. 1982. Making and correcting errors during sentence comprehension: Eye movements in the analysis of structurally ambiguous sentences. *Cognitive Psychology*, 14(2):178–210.
- Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of the 24th International Conference on Computational Linguistics (Coling 2012)*. Association for Computational Linguistics, Mumbai, India.
- Matthew Honnibal, Yoav Goldberg, and Mark Johnson. 2013. A non-monotonic arc-eager transition system for dependency parsing. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 163–172. Association for Computational Linguistics, Sofia, Bulgaria.
- Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–151. Association for Computational Linguistics, Montréal, Canada.
- Mark Johnson and Eugene Charniak. 2004. A TAG-based noisy channel model of speech repairs. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 33–39.
- Douglas A. Jones, Florian Wolf, Edward Gibson, Elliott Williams, Evelina Fedorenko, Douglas A. Reynolds, and Marc A. Zissman. 2003. Measuring the readability of automatic speech-to-text transcripts. In *INTERSPEECH*. ISCA.
- Fredrik Jorgensen. 2007. The effects of disfluency detection in parsing spoken language. In Joakim Nivre, Heiki-Jaan Kaalep, Kadri Muischnek, and Mare Koit, editors, *Proceedings of the 16th Nordic Conference of Computational Linguistics NODALIDA-2007*, pages 240–244.
- Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1–11.
- Percy Liang. 2005. *Semi-supervised learning for natural language*. Ph.D. thesis, MIT.
- Christopher D. Manning. 2011. Part-of-speech tagging from 97linguistics? In *Proceedings of the 12th international conference on Computational linguistics and intelligent text processing - Volume Part I, CICLing’11*, pages 171–189. Springer-Verlag, Berlin, Heidelberg.
- 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.
- Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings*

- of the 8th International Workshop on Parsing Technologies (IWPT), pages 149–160.
- Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34:513–553.
- Xian Qian and Yang Liu. 2013. Disfluency detection using multi-step stacked learning. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 820–825. Association for Computational Linguistics, Atlanta, Georgia.
- Mohammad Sadegh Rasooli and Joel Tetreault. 2013. Joint parsing and disfluency detection in linear time. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 124–129. Association for Computational Linguistics, Seattle, Washington, USA.
- Brian Roark, Mary Harper, Eugene Charniak, Bonnie Dorr, Mark Johnson, Jeremy Kahn, Yang Liu, Mary Ostendorf, John Hale, Anna Krasnyanskaya, Matthew Lease, Izhak Shafran, Matthew Snover, Robin Stewart, and Lisa Yung. 2006. Sparseval: Evaluation metrics for parsing speech. In *Proceedings of Language Resource and Evaluation Conference*, pages 333–338. European Language Resources Association (ELRA), Genoa, Italy.
- Francesco Sartorio, Giorgio Satta, and Joakim Nivre. 2013. A transition-based dependency parser using a dynamic parsing strategy. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 135–144. Association for Computational Linguistics, Sofia, Bulgaria.
- Elizabeth Shriberg. 1994. *Preliminaries to a Theory of Speech Disfluencies*. Ph.D. thesis, University of California, Berkeley.
- Xu Sun, Takuya Matsuzaki, Daisuke Okanohara, and Jun'ichi Tsujii. 2009. Latent variable perceptron algorithm for structured classification. In *IJCAI*, pages 1236–1242.
- Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics, Uppsala, Sweden.
- Dongdong Zhang, Shuangzhi Wu, Nan Yang, and Mu Li. 2013. Punctuation prediction with transition-based parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 752–760. Association for Computational Linguistics, Sofia, Bulgaria.
- Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Computational Linguistics*, 37(1):105–151.
- Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193. Association for Computational Linguistics, Portland, USA.
- Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 434–443. Association for Computational Linguistics, Sofia, Bulgaria.
- Simon Zwarts and Mark Johnson. 2011. The impact of language models and loss functions on repair disfluency detection. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 703–711. Association for Computational Linguistics, Portland, USA.
- Simon Zwarts, Mark Johnson, and Robert Dale. 2010. Detecting speech repairs incrementally using a noisy channel approach. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1371–1378. Coling 2010 Organizing Committee, Beijing, China.

A Crossing-Sensitive Third-Order Factorization for Dependency Parsing

Emily Pitler*

Google Research

76 9th Avenue

New York, NY 10011

epitler@google.com

Abstract

Parsers that parametrize over wider scopes are generally more accurate than edge-factored models. For graph-based non-projective parsers, wider factorizations have so far implied large increases in the computational complexity of the parsing problem. This paper introduces a “crossing-sensitive” generalization of a third-order factorization that trades off complexity in the *model* structure (i.e., scoring with features over multiple edges) with complexity in the *output* structure (i.e., producing crossing edges). Under this model, the optimal 1-Endpoint-Crossing tree can be found in $O(n^4)$ time, matching the asymptotic run-time of *both* the third-order *projective* parser and the *edge-factored* 1-Endpoint-Crossing parser. The crossing-sensitive third-order parser is significantly more accurate than the third-order projective parser under many experimental settings and significantly less accurate on none.

1 Introduction

Conditioning on wider syntactic contexts than simply individual head-modifier relationships improves parsing accuracy in a wide variety of parsers and frameworks (Charniak and Johnson, 2005; McDonald and Pereira, 2006; Hall, 2007; Carreras, 2007; Martins et al., 2009; Koo and Collins, 2010; Zhang and Nivre, 2011; Bohnet and Kuhn, 2012; Martins et al., 2013). This paper proposes a new graph-based dependency parser that efficiently produces

the globally optimal dependency tree according to a third-order model (that includes features over grandparents and siblings in the tree) in the class of 1-Endpoint-Crossing trees (that includes all projective trees and the vast majority of *non-projective* structures seen in dependency treebanks).

Within graph-based *projective* parsing, the *third-order* parser of Koo and Collins (2010) has a run-time of $O(n^4)$, just one factor of n more expensive than the edge-factored model of Eisner (2000). Incorporating richer features *and* producing trees with crossing edges has traditionally been a challenge, however, for graph-based dependency parsers. If parsing is posed as the problem of finding the optimal scoring directed spanning tree, then the problem becomes NP-hard when trees are scored with a grandparent and/or sibling factorization (McDonald and Pereira, 2006; McDonald and Satta, 2007). For various definitions of mildly non-projective trees, even edge-factored versions are expensive, with *edge-factored* running times between $O(n^4)$ and $O(n^7)$ (Gómez-Rodríguez et al., 2011; Pitler et al., 2012; Pitler et al., 2013; Satta and Kuhlmann, 2013).

The third-order projective parser of Koo and Collins (2010) and the edge-factored 1-Endpoint-Crossing parser described in Pitler et al. (2013) have some similarities: both use $O(n^4)$ time and $O(n^3)$ space, using sub-problems over intervals with one exterior vertex, which are constructed using one free split point. The two parsers differ in *how the exterior vertex is used*: Koo and Collins (2010) use the exterior vertex to store a grandparent index, while Pitler et al. (2013) use the exterior vertex to introduce crossed edges between the point and

*The majority of this work was done while at the University of Pennsylvania.

	Projective	1-Endpoint-Crossing
Edge	$O(n^3)$ Eisner (2000)	$O(n^4)$ Pitler et al. (2013)
CS-GSib	$O(n^4)$ Koo and Collins (2010)	$O(n^4)$ This paper

Table 1: Parsing time for various output spaces and model factorizations. CS-GSib refers to the (crossing-sensitive) grand-sibling factorization described in this paper.

the interval. This paper proposes *merging* the two parsers to achieve the best of both worlds – producing the best tree in the wider range of 1-Endpoint-Crossing trees while incorporating the identity of the grandparent and/or sibling of the child in the score of an edge whenever the local neighborhood of the edge does not contain crossing edges. The crossing-sensitive grandparent-sibling 1-Endpoint-Crossing parser proposed here takes $O(n^4)$ time, matching the runtime of both the third-order projective parser and of the edge-factored 1-Endpoint-Crossing parser (see Table 1).

The parsing algorithms of Koo and Collins (2010) and Pitler et al. (2013) are reviewed in Section 2. The proposed crossing-sensitive factorization is defined in Section 3. The parsing algorithm that finds the optimal 1-Endpoint-Crossing tree according to this factorization is described in Section 4. The implemented parser is significantly more accurate than the third-order projective parser in a variety of languages and treebank representations (Section 5). Section 6 discusses the proposed approach in the context of prior work on non-projective parsing.

2 Preliminaries

In a *projective* dependency tree, each subtree forms one consecutive interval in the sequence of input words; equivalently (assuming an artificial root node placed as either the first or last token), when all edges are drawn in the half-plane above the sentence, no two edges *cross* (Kübler et al., 2009). Two vertex-disjoint edges *cross* if their endpoints interleave. A *1-Endpoint-Crossing* tree is a dependency tree such that for each edge, all edges that cross it share a common vertex (Pitler et al., 2013). Note that the class of projective trees is properly included within the class of 1-Endpoint-Crossing trees.

To avoid confusion between intervals and edges,

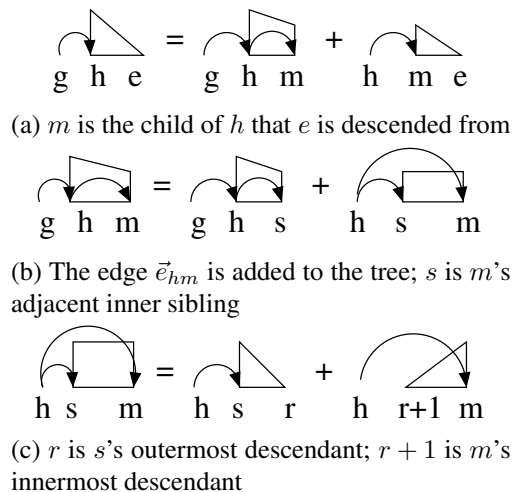


Figure 1: Algorithm for grand-sibling projective parsing; the figures replicate Figure 6 in Koo and Collins (2010).

\vec{e}_{ij} denotes the *directed edge* from i to j (i.e., i is the parent of j). Interval notation $((i, j), [i, j], (i, j],$ or $[i, j))$ is used to denote *sets of vertices* between i and j , with square brackets indicating closed intervals and round brackets indicating open intervals.

2.1 Grand-Sibling Projective Parsing

A grand-sibling factorization allows features over 4-tuples of (g, h, m, s) , where h is the parent of m , g is m 's grandparent, and s is m 's adjacent inner sibling. Features over these grand-sibling 4-tuples are referred to as “third-order” because they scope over three *edges* simultaneously (\vec{e}_{gh} , \vec{e}_{hs} , and \vec{e}_{hm}). The parser of Koo and Collins (2010) produces the highest-scoring projective tree according to this grand-sibling model by adding an external grandparent index to each of the sub-problems used in the sibling factorization (McDonald and Pereira, 2006). Figure 6 in Koo and Collins (2010) provided a pictorial view of the algorithm; for convenience, it is replicated in Figure 1. An edge \vec{e}_{hm} is added to the tree in the “trapezoid” step (Figure 1b); this allows the edge to be scored conditioned on m 's grandparent (g) and its adjacent inner sibling (s), as all four relevant indices are accessible.

2.2 Edge-factored 1-Endpoint-Crossing Parsing

The edge-factored 1-Endpoint-Crossing parser of Pitler et al. (2013) produces the highest scoring 1-

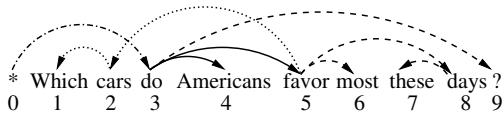


Figure 2: A 1-Endpoint-Crossing non-projective English sentence from the WSJ Penn Treebank (Marcus et al., 1993), converted to dependencies with PennConverter (Johansson and Nugues, 2007).

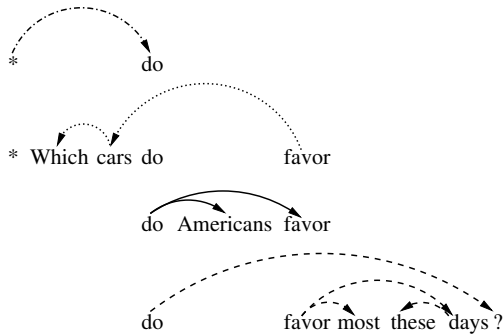


Figure 3: Constructing a 1-Endpoint-Crossing tree with intervals with one exterior vertex (Pitler et al., 2013).

Endpoint-Crossing tree with each edge \vec{e}_{hm} scored according to $\text{Score}(\text{Edge}(h, m))$. The 1-Endpoint-Crossing property allows the tree to be built up in edge-disjoint pieces each consisting of intervals with one exterior point that has edges into the interval. For example, the tree in Figure 2 would be built up with the sub-problems shown in Figure 3.

To ensure that crossings *within* a sub-problem are consistent with the crossings that happen as a result of combination steps, the algorithm uses four different “types” of sub-problems, indicating whether the edges incident to the exterior point may be internally crossed by edges incident to the left boundary point (L), the right (R), either (LR), or neither (N). In Figure 3, the sub-problem over $[*, do] \cup \{favor\}$ would be of type R , and $[favor, ?] \cup \{do\}$ of type L .

2.2.1 Naïve Approach to Including Grandparent Features

The example in Figure 3 illustrates the difficulty of incorporating grandparents into the scoring of all edges in 1-Endpoint-Crossing parsing. The vertex *favor* has a parent or child in *all three* of the sub-problems. In order to use grandparent scoring for the edges from *favor* to *favor*’s children in the other two sub-problems, we would need to augment the problems with the grandparent index *do*. We also

must add the parent index *do* to the middle sub-problem to ensure consistency (i.e., that *do* is in fact the parent assigned). Thus, a first attempt to score all edges with grandparent features within 1-Endpoint-Crossing trees raises the runtime from $O(n^4)$ to $O(n^7)$ (all of the four indices need a “predicted parent” index; at least one edge is always implied so one of these additional indices can be dropped).

3 Crossing-Sensitive Factorization

Factorizations for projective dependency parsing have often been designed to allow efficient parsing. For example, the algorithms in Eisner (2000) and McDonald and Pereira (2006) achieve their efficiency by assuming that children to the left of the parent and to the right of the parent are independent of each other. The algorithms of Carreras (2007) and Model 2 in Koo and Collins (2010) include grandparents for only the outermost grand-children of each parent for efficiency reasons. In a similar spirit, this paper introduces a variant of the Grand-Sib factorization that scores crossed edges independently (as a *CrossedEdge* part) and uncrossed edges under either a grandparent-sibling, grandparent, sibling, or edge-factored model depending on whether relevant edges in its local neighborhood are crossed.

A few auxiliary definitions are required. For any parent h and grandparent g , h ’s children are partitioned into *interior* children (those between g and h) and *exterior children* (the complementary set of children).¹ Interior children are numbered from closest to h through furthest from h ; exterior children are first numbered on the side closer to h from closest to h through furthest, then the enumeration wraps around to include the vertices on the side closer to g . Figure 4 shows a parent h , its grandparent g , and a possible sequence of three interior and four exterior children. Note that for a projective tree, there would not be any children on the far side of g .

Definition 1. Let h be m ’s parent. $\text{Outer}(m)$ is the set of siblings of m that are in the same subset of h ’s children and are later in the enumeration than m is.

For example, in the tree in Figure 2,

¹Because dependency trees are directed trees, each node except for the artificial root has a unique parent. To ensure that *grandparent* is defined for the root’s children, assume an artificial parent of the root for notational convenience.

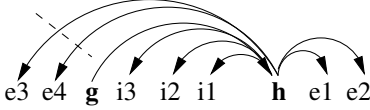


Figure 4: The exterior children are numbered first beginning on the side closest to the parent, then the side closest to the grandparent. There must be a path from the root to g , so the edges from h to its exterior children on the far side of g are guaranteed to be crossed.

	$Crossed(\vec{e}_{hs})$	$\neg Crossed(\vec{e}_{hs})$
$\neg GProj(\vec{e}_{hm})$	Edge(h, m)	Sib(h, m, s)
$GProj(\vec{e}_{hm})$	Grand(g, h, m)	GrandSib(g, h, m, s)

Table 2: Part type for an uncrossed edge \vec{e}_{hm} for the crossing-sensitive third-order factorization (g is m 's grandparent; s is m 's inner sibling).

$$Outer(most) = \{days, cars\}.$$

Definition 2. An uncrossed edge \vec{e}_{hm} is $GProj$ if both of the following hold:

1. The edge \vec{e}_{gh} from h 's parent to h is not crossed
2. None of the edges from h to $Outer(m)$ (m 's outer siblings) are crossed

Uncrossed $GProj$ edges include the grandparent in the part. The part includes the sibling if the edge \vec{e}_{hs} from the parent to the sibling is not crossed. Table 2 gives the factorization for uncrossed edges.

The parser in this paper finds the optimal 1-Endpoint-Crossing tree according to this factorized form. A fully projective tree would decompose into *exclusively* GrandSib parts (as all edges would be uncrossed and $GProj$). As all projective trees are within the 1-Endpoint-Crossing search space, the optimization problem that the parser solves includes all projective trees scored with grand-sibling features everywhere. Projective parsing with grand-sibling scores can be seen as a special case, as the crossing-sensitive 1-Endpoint-Crossing parser can simulate a grand-sibling projective parser by setting all $Crossed(h, m)$ scores to $-\infty$.

In Figure 2, the edge from do to $Americans$ is not $GProj$ because Condition (1) is violated, while the edge from $favor$ to $most$ is not $GProj$ because Condition (2) is violated. Under this definition, the vertices do and $favor$ (which have children in multiple sub-problems) do not need external grandparent indices in *any* of their sub-problems. Table 3

CrossedEdge(*, do)	Sib($cars, Which, -$)
CrossedEdge($favor, cars$)	Sib($do, Americans, -$)
Sib($do, favor, Americans$)	CrossedEdge($do, ?$)
Sib($favor, most, -$)	Sib($favor, days, most$)
GSib($favor, days, these, -$)	

Table 3: Decomposing Figure 2 according to the crossing-sensitive third-order factorization described in Section 3. Null inner siblings are indicated with $-$.

lists the parts in the tree in Figure 2 according to this crossing-sensitive third-order factorization.

4 Parsing Algorithm

The parser finds the maximum scoring 1-Endpoint-Crossing tree according to the factorization in Section 3 with a dynamic programming procedure reminiscent of Koo and Collins (2010) (for scoring uncrossed edges with grandparent and/or sibling features) and of Pitler et al. (2013) (for including crossed edges). The parser also uses novel sub-problems for transitioning between portions of the tree with and without crossed edges. This formulation of the parsing problem presents two difficulties:

1. The parser must know whether an edge is crossed when it is added.
2. For *uncrossed* edges, the parser must use the appropriate part for scoring according to whether *other* edges are crossed (Table 2).

Difficulty 1 is solved by adding crossed and uncrossed edges to the tree in distinct sub-problems (Section 4.1). Difficulty 2 is solved by producing different versions of subtrees over the same sets of vertices, both with and without a grandparent index, which differ in their assumptions about the tree *outside* of that set (Section 4.2). The list of all sub-problems with their invariants and the full dynamic program are provided in the supplementary material.

4.1 Enforcing Crossing Edges

The parser adds crossed and uncrossed edges in distinct portions of the dynamic program. Uncrossed edges are added *only* through trapezoid sub-problems (that may or may not have a grandparent index), while crossed edges are added in *non*-trapezoid sub-problems. To add *all* uncrossed edges

in trapezoid sub-problems, the parser (a) enforces that any edge added anywhere else must be crossed, and (b) includes transitional sub-problems to build trapezoids when the edge \vec{e}_{hm} is not crossed, but the edge to its inner sibling \vec{e}_{hs} is (and so the construction step shown in Figure 1b cannot be used).

4.1.1 Crossing Conditions

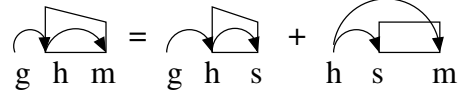
Pitler et al. (2013) included crossing edges by using “crossing region” sub-problems over intervals with an external vertex that *optionally* contained edges between the interval and the external vertex. An uncrossed edge could then be included either by a derivation that prohibited it from being crossed or a derivation which allowed (but did not force) it to be crossed. This ambiguity is removed by enforcing that (1) each crossing region contains at least one edge incident to the exterior vertex, and (2) all such edges are crossed by edges in another sub-problem. For example, by requiring at least one edge between *do* and *(favor, ?]* and also between *favor* and *(*, do)*, the edges in the two sets are guaranteed to cross.

4.1.2 Trapezoids with Edge to Inner Sibling Crossed

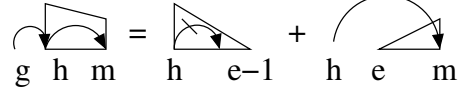
To add *all* uncrossed edges in trapezoid-style sub-problems, we must be able to construct a trapezoid over vertices $[h, m]$ whenever the edge \vec{e}_{hm} is not crossed. The construction used in Koo and Collins (2010), repeated graphically in Figure 5a, *cannot* be used if the edge \vec{e}_{hs} is crossed, as there would then exist edges between (h, s) and (s, m) , making s an invalid split point. The parser therefore includes some “transitional glue” to allow alternative ways to construct the trapezoid over $[h, m]$ when \vec{e}_{hm} is not crossed but the edge \vec{e}_{hs} to m ’s inner sibling is.

The two additional ways of building trapezoids are shown graphically in Figures 5b and 5c. Consider the “chain of crossing edges” that includes the edge \vec{e}_{hs} . If none of these edges are in the subtree rooted at m , then we can build the tree involving m and its inner descendants separately (Figure 5b) from the rest of the tree rooted at h . Within the interval $[h, e - 1]$ the furthest edge incident to h (\vec{e}_{hs}) must be crossed: these intervals are parsed choosing s and the crossing point of \vec{e}_{hs} simultaneously (as in Figure 4 in Pitler et al. (2013)).

Otherwise, the sub-tree rooted at m is involved in



(a) Edge from h to inner sibling s is *not* crossed (repeats Figure 1b)



(b) \vec{e}_{hs} is crossed, but the chain of crossing edges involving \vec{e}_{hs} does not include any descendants of m . e is m ’s descendant furthest from m within (h, m) . $s \in (h, e - 1)$.



(c) \vec{e}_{hs} is crossed, and the chain of crossing edges involving \vec{e}_{hs} includes descendants of m . Of m ’s descendants that are incident to edges in the chain, d is the one closest to m (d can be m itself). $s \in (h, d)$.

Figure 5: Ways to build a trapezoid when the edge \vec{e}_{hs} to m ’s inner sibling may be crossed.

the chain of crossing edges (Figure 5c). The chain of crossing edges between h and d (m ’s descendant, which may be m itself) is built up first then concatenated with the triangle rooted at m containing m ’s inner descendants not involved in the chain.

Chains of crossing edges are constructed by repeatedly applying two specialized types of L items that alternate between adding an edge from the interval to the exterior point (right-to-left) or from the exterior point to the interval (left-to-right) (Figure 6). The boundary edges of the chain can be crossed more times without violating the 1-Endpoint-Crossing property, and so the beginning and end of the chain can be unrestricted crossing regions. These specialized chain sub-problems are also used to construct *boxes* (Figure 1c) over $[s, m]$ with shared parent h when neither edge \vec{e}_{hs} nor \vec{e}_{hm} is crossed, but the subtrees rooted at m and at s cross each other (Figure 7).

Lemma 1. *The GrandSib-Crossing parser adds all uncrossed edges and only uncrossed edges in a tree in a “trapezoid” sub-problem.*

Proof. The *only* part is easy: when a trapezoid is built over an interval $[h, m]$, all edges are internal to the interval, so no earlier edges could cross \vec{e}_{hm} . Af-

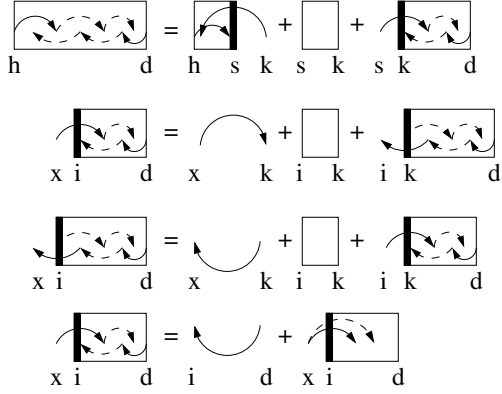


Figure 6: Constructing a chain of crossing edges

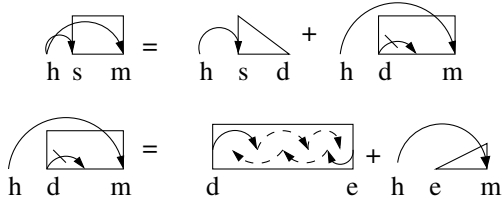


Figure 7: Constructing a box when edges in m and s 's subtrees cross each other.

ter the trapezoid is built, only the interval endpoints h and m are accessible for the rest of the dynamic program, and so an edge between a vertex in (h, m) and a vertex $\notin [h, m]$ can never be added. The Crossing Conditions ensure that every edge added in a non-trapezoid sub-problem is crossed. \square

Lemma 2. *The GrandSib-Crossing parser considers all 1-Endpoint-Crossing trees and only 1-Endpoint-Crossing trees.*

Proof. All trees that could have been built in Pitler et al. (2013) are still possible. It can be verified that the additional sub-problems added all obey the 1-Endpoint-Crossing property. \square

4.2 Reduced Context in Presence of Crossings

A crossed edge (added in a non-trapezoid sub-problem) is scored as a CrossedEdge part. An uncrossed edge added in a trapezoid sub-problem, however, may need to be scored according to a GrandSib, Grand, Sib, or Edge part, depending on whether the relevant *other* edges are crossed. In this section we show that sibling and grandparent features are included in the GrandSib-Crossing parser as specified by Table 2.

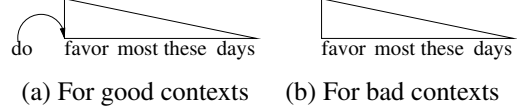


Figure 8: For each of the interval sub-problems in Koo and Collins (2010), the parser constructs versions with and without the additional grandparent index. Figure 8b is used if the edge from *do* to *favor* is crossed, or if there are any crossed edges from *favor* to children to the left of *do* or to the right of *days*. Otherwise, Figure 8a is used.

4.2.1 Sibling Features

Lemma 3. *The GrandSib-Crossing parser scores an uncrossed edge \vec{e}_{hm} with a Sib or GrandSib part if and only if \vec{e}_{hs} is not crossed.*

Proof. Whether the edge to an uncrossed edge's inner sibling is crossed is known bottom-up through how the trapezoid is constructed, since the inner sibling is *internal* to the sub-problem. When \vec{e}_{hs} is not crossed, the trapezoid is constructed as in Figure 5a, using the inner sibling as the split point. When the edge \vec{e}_{hs} is crossed, the trapezoid is constructed as in Figure 5b or 5c; note that both ways force the edge to the inner sibling to be crossed. \square

4.2.2 Grandparent Features for $GProj$ Edges

Koo and Collins (2010) include an external grandparent index for each of the sub-problems that the edges within use for scoring. We want to avoid adding such an external grandparent index to *any* of the crossing region sub-problems (to stay within the desired time and space constraints) or to interval sub-problems when the external context would make all internal edges $\neg GProj$.

For each interval sub-problem, the parser constructs versions both with and without a grandparent index (Figure 8). Which version is used depends on the external context. In a *bad context*, all edges to children within an interval are guaranteed to be $\neg GProj$. This section shows that all boundary points in crossing regions are placed in bad contexts, and then that edges are scored with grandparent features if and only if they are $GProj$.

Bad Contexts for Interval Boundary Points For *exterior vertex* boundary points, all edges from it to its children will be crossed (Section 4.1.1), so it does not need a grandparent index.

Lemma 4. *If a boundary point i 's parent (call it g) is within a sub-problem over vertices $[i, j]$ or $[i, j] \cup \{x\}$, then for all uncrossed edges \vec{e}_{im} with m in the sub-problem, the tree outside of the sub-problem is irrelevant to whether \vec{e}_{im} is $GProj$.*

Proof. The sub-problem contains the edge \vec{e}_{gi} , so Condition (1) is checked internally. m cannot be x , since \vec{e}_{im} is uncrossed. If g is x , then \vec{e}_{im} is $\neg GProj$ regardless of the outer context. If both g and $m \in (i, j]$, then $Outer(m) \subseteq (i, j]$: If m is an interior child of i ($m \in (i, g)$) then $Outer(m) \subseteq (m, g) \subseteq (i, j]$. Otherwise, if m is an exterior child ($m \in (g, j]$), by the ‘‘wrapping around’’ definition of $Outer$, $Outer(m) \subseteq (g, m) \subseteq (i, j]$. Thus Condition (2) is also checked internally. \square

We can therefore focus on interval boundary points with their parent outside of the sub-problem.

Definition 3. *The left boundary vertex of an interval $[i, j]$ is in a bad context ($BadContext_L(i, j)$) if i receives its parent (call it g) from outside of the sub-problem and either of the following hold:*

1. **Grand-Edge Crossed:** \vec{e}_{gi} is crossed
2. **Outer-Child-Edge Crossed:** *An edge from i to a child of i outside of $[i, j]$ and Outer to j will be crossed (recall this includes children on the far side of g if g is to the left of i)*

$BadContext_R(i, j)$ is defined symmetrically regarding j and j 's parent and children.

Corollary 1. *If $BadContext_L(i, j)$, then for all \vec{e}_{im} with $m \in (i, j]$, \vec{e}_{im} is $\neg GProj$. Similarly, if $BadContext_R(i, j)$, for all \vec{e}_{jm} with $m \in [i, j)$, \vec{e}_{jm} is $\neg GProj$.*

No Grandparent Indices for Crossing Regions

We would exceed the desired $O(n^4)$ run-time if any crossing region sub-problems needed any grandparent indices. In Pitler et al. (2013), LR sub-problems with edges from the exterior point crossed by both the left and the right boundary points were constructed by concatenating an L and an R sub-problem. Since the split point was not necessarily incident to a crossed edge, the split point might have $GProj$ edges to children on the side other than where it gets its parent; accommodating this would add another factor of n to the running time and space

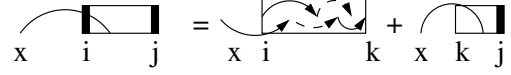


Figure 9: For all split points k , the edge from k 's parent to k is crossed, so all edges from k to children on either side were $\neg GProj$. The case when the split point's parent is from the right is symmetric.



(a) x is Outer to all children of k in $(k, j]$. (b) x is Outer to all children of k in $[i, k)$.

Figure 10: The edge \vec{e}_{kx} is guaranteed to be crossed, so k is in a $BadContext$ for whichever side it does not get its parent from.

to store the split point's parent. To avoid this increase in running time, they are instead built up as in Figure 9, which chooses the split point so that the edge from the parent of the split point to it is crossed.

Lemma 5. *For all crossing region sub-problems $[i, j] \cup \{x\}$ with i 's parent $\notin [i, j] \cup \{x\}$, $BadContext_L(i, j)$. Similarly, when j 's parent $\notin [i, j] \cup \{x\}$, $BadContext_R(i, j)$.*

Proof. Crossing region sub-problems either combine to form intervals or larger crossing regions. When they combine to form intervals as in Figure 3, it can be verified that all boundary points are in a bad context. LR sub-problems were discussed above. Split points for the $L/R/N$ sub-problems by construction are incident to a crossed edge to a further vertex. If that edge is from the split point's parent to the split point, then the grand-edge is crossed and so both sides are in a bad context. If the crossed edge is from the split point to a child, then that child is Outer to all other children on the side in which it does not get its parent (see Figure 10). \square

Corollary 2. *No grandparent indices are needed for any crossing region sub-problem.*

Triangles and Trapezoids with and without Grandparent Indices

The presentation that follows assumes left-headed versions. Uncrossed edges are added in two distinct types of trapezoids: (1) $TrapG[h, m, g, L]$ with an external grandparent index g , scores the edge \vec{e}_{hm} with grandpar-

ent features, and (2) $\text{Trap}[h, m, L]$ without a grandparent index, scores the edge \vec{e}_{hm} without grandparent features. Triangles also have versions with ($\text{TriG}[h, e, g, L]$) and without ($\text{Tri}[h, e, L]$) a grandparent index. What follows shows that all $G\text{Proj}$ edges are added in TrapG sub-problems, and all $\neg G\text{Proj}$ uncrossed edges are added in Trap sub-problems.

Lemma 6. *For all $k \in (i, j)$, if $\text{BadContext}_L(i, j)$, then $\text{BadContext}_L(i, k)$. Similarly, if $\text{BadContext}_R(i, j)$, then $\text{BadContext}_R(k, j)$.*

Proof. $\text{BadContext}_L(i, j)$ implies either the edge from i 's parent to i is crossed and/or an edge from i to a child of i outer to j is crossed. If the edge from i 's parent to i is crossed, then $\text{BadContext}_L(i, k)$. If a child of i is outer to j , then since $k \in (i, j)$, such a child is also outer to k . \square

Lemma 7. *All left-rooted triangle sub-problems $\text{Tri}[i, j, L]$ without a grandparent index are in a $\text{BadContext}_L(i, j)$. Similarly for all $\text{Tri}[i, j, R]$, $\text{BadContext}_R(i, j)$.*

Proof. All triangles without grandparent indices are either placed immediately into a bad context (by adding a crossed edge to the triangle's root from its parent, or a crossed edge from the root to an outer child) or are combined with other sub-trees to form larger crossing regions (and therefore the triangle is in a bad context, using Lemmas 5 and 6). \square

Lemma 8. *All triangle sub-problems with a grandparent index $\text{TriG}[h, e, g, L]$ are placed in a $\neg \text{BadContext}_L(h, e)$. Similarly, $\text{TriG}[e, h, g, R]$ are only placed in $\neg \text{BadContext}_R(h, e)$.*

Proof. Consider where a non-empty triangle ($h \neq e$) with a grandparent index $\text{TriG}[h, e, g, L]$ can be placed in the full dynamic program and what each step would imply about the rest of the tree.

If the triangle contains exterior children of h (e and g are on opposite sides of h), then it can either combine with a trapezoid to form another larger triangle (as in Figure 1a) or it can combine with another sub-problem to form a box with a grandparent index (Figure 1c or 7). Boxes with a grandparent index can only combine with another trapezoid to form a larger trapezoid (Figure 1b). Both cases

force \vec{e}_{gh} to not be crossed and prevent h from having any outer crossed children, as h becomes an internal node within the larger sub-problem.

If the triangle contains interior children of h (e lies between g and h), then it can either form a trapezoid from g to h by combining with a triangle (Figure 5b) or a chain of crossing edges (Figure 5c), or it can be used to build a box with a grandparent index (Figures 1c and 7), which then can only be used to form a trapezoid from g to h . In either case, a trapezoid is constructed from g to h , enforcing that \vec{e}_{gh} cannot be crossed. These steps prevent h from having any additional children between g and e (since h does not appear in the adjacent sub-problems at all whenever $h \neq e$), so again the children of h in (e, h) have no outer siblings. \square

Lemma 9. *In a $\text{TriG}[h, e, g, L]$ sub-problem, if an edge \vec{e}_{hm} is not crossed and no edges from i to siblings of m in $(m, e]$ are crossed, then \vec{e}_{hm} is $G\text{Proj}$.*

Proof. This follows from (1) the edge \vec{e}_{hm} is not crossed, (2) the edge \vec{e}_{gh} is not crossed by Lemma 8, and (3) no outer siblings are crossed (outer siblings in $(m, e]$ are not crossed by assumption and siblings outer to e are not crossed by Lemma 8). \square

Lemma 10. *An edge \vec{e}_{hm} scored with a GrandSib or Grand part (added through a $\text{TrapG}[h, m, g, L]$ or $\text{TrapG}[m, h, g, R]$ sub-problem) is $G\text{Proj}$.*

Proof. A TrapG can either (1) combine with descendants of m to form a triangle with a grandparent index rooted at h (indicating that m is the outermost inner child of h) or (2) combine with descendants of m and of m 's adjacent outer sibling (call it o), forming a trapezoid from h to o (indicating that \vec{e}_{ho} is not crossed). Such a trapezoid could again only combine with further uncrossed outer siblings until eventually the final triangle rooted at h with grandparent index g is built. As \vec{e}_{hm} was not crossed, no edges from h to outer siblings within the triangle are crossed, and \vec{e}_{hm} is within a TriG sub-problem, \vec{e}_{hm} is $G\text{Proj}$ by Lemma 9. \square

Lemma 11. *An uncrossed edge \vec{e}_{hm} scored with a Sib or Edge part (added through a $\text{Trap}[h, m, L]$ or $\text{Trap}[m, h, R]$ sub-problem) is $\neg G\text{Proj}$.*

Proof. A Trap can only (1) form a triangle without a grandparent index, or (2) form a trapezoid to an outer sibling of m , until eventually a final triangle rooted at h without a grandparent index is built. This triangle without a grandparent index is then placed in a bad context (Lemma 7) and so \vec{e}_{hm} is $\neg GProj$ (Corollary 1). \square

4.3 Main Results

Lemma 12. *The crossing-sensitive third-order parser runs in $O(n^4)$ time and $O(n^3)$ space when the input is an unpruned graph. When the input to the parser is a pruned graph with at most k incoming edges per node, the crossing-sensitive third-order parser runs in $O(kn^3)$ time and $O(n^3)$ space.*

Proof. All sub-problems are either over intervals (two indices), intervals with a grandparent index (three indices), or crossing regions (three indices). No crossing regions require any grandparent indices (Corollary 2). The only sub-problems that require a maximization over two internal split points are over intervals and need no grandparent indices (as the furthest edges from each root are guaranteed to be crossed within the sub-problem). All steps either contain an edge in their construction step or in the invariant of the sub-problem, so with a pruned graph as input, the running time is the number of edges ($O(kn)$) times the number of possibilities for the other two free indices ($O(n^2)$). The space is not reduced as there is not necessarily an edge relationship between the three stored vertices. \square

Theorem 1. *The GrandSib-Crossing parser correctly finds the maximum scoring 1-Endpoint-Crossing tree according to the crossing-sensitive third-order factorization (Section 3) in $O(n^4)$ time and $O(n^3)$ space. When the input to the parser is a pruned graph with at most k incoming edges per node, the GrandSib-Crossing parser correctly finds the maximum scoring 1-Endpoint-Crossing tree that uses only unpruned edges in $O(kn^3)$ time and $O(n^3)$ space.*

Proof. The correctness of scoring follows from Lemmas 3, 10, and 11. The search space of 1-Endpoint-Crossing trees was in Lemma 2 and the time and space complexity in Lemma 12. \square

The parser produces the optimal tree in a well-defined output space. Pruning edges restricts the output space the same way that constraints enforcing projectivity or the 1-Endpoint-Crossing property also restrict the output space. Note that if the optimal unconstrained 1-Endpoint-Crossing tree does not include any pruned edges, then whether the parser uses pruning or not is irrelevant; both the pruned and unpruned parsers will produce the exact same tree.

5 Experiments

The crossing-sensitive third-order parser was implemented as an alternative parsing algorithm within *dpo3* (Koo and Collins, 2010).² To ensure a fair comparison, all code relating to input/output, features, learning, etc. was re-used from the original projective implementation, and so the only substantive differences between the projective and 1-Endpoint-Crossing parsers are the dynamic programming charts, the parsing algorithms, and the routines that extract the maximum scoring tree from the completed chart.

The treebanks used to prepare the CoNLL shared task data (Buchholz and Marsi, 2006; Nivre et al., 2007) vary widely in their conventions for representing conjunctions, modal verbs, determiners, and other decisions (Zeman et al., 2012). The experiments use the newly released HamleDT software (Zeman et al., 2012) that normalizes these treebanks into one standard format and also provides built-in transformations to other conjunction styles. The unnormalized treebanks input to HamleDT were from the CoNLL 2006 Shared Task (Buchholz and Marsi, 2006) for Danish, Dutch, Portuguese, and Swedish and from the CoNLL 2007 Shared Task (Nivre et al., 2007) for Czech.

The experiments include the default Prague style (Böhmová et al., 2001), Mel’čukian style (Mel’čuk, 1988), and Stanford style (De Marneffe and Manning, 2008) for conjunctions. Under the grandparent-sibling factorization, the two words being conjoined would never appear in the same scope for the Prague style (as they are siblings on different sides of the conjunct head). In the Mel’čukian style, the two conjuncts are in a grandparent relationship and in the Stanford style the two conjuncts

²<http://groups.csail.mit.edu/nlp/dpo3/>

are in a sibling relationship, and so we would expect to see larger gains for including grandparents and siblings under the latter two representations. The experiments also include a nearly projective dataset, the English Penn Treebank (Marcus et al., 1993), converted to dependencies with PennConverter (Johansson and Nugues, 2007).

The experiments use marginal-based pruning based on an edge-factored directed spanning tree model (McDonald et al., 2005). Each word’s set of potential parents is limited to those with a marginal probability of at least .1 times the probability of the most probable parent, and cut off this list at a maximum of 20 potential parents per word. To ensure that there is always at least one projective and/or 1-Endpoint-Crossing tree achievable, the artificial root is always included as an option. The pruning parameters were chosen to keep 99.9% of the true edges on the English development set.

Following Carreras (2007) and Koo and Collins (2010), before training the training set trees are transformed to be the best achievable within the model class (i.e., the closest projective tree or 1-Endpoint-Crossing tree). All models are trained for five iterations of averaged structured perceptron training. For English, the model after the iteration that performs best on the development set is used; for all other languages, the model after the fifth iteration is used.

5.1 Results

Results for edge-factored and (crossing-sensitive) grandparent-sibling factored models for both projective and 1-Endpoint-Crossing parsing are in Tables 4 and 5. In 14 out of the 16 experimental set-ups, the third-order 1-Endpoint-Crossing parser is more accurate than the third-order projective parser. It is significantly better than the projective parser in 9 of the set-ups and significantly worse in none.

Table 6 shows how often the 1-EC CS-GSib parser used each of the GrandSib, Grand, Sib, Edge, and CrossedEdge parts for the Mel’čukian and Stanford style test sets. In both representations,

³Following prior work in graph-based dependency parsing (for example, Rush and Petrov (2012)), English results use automatically produced part-of-speech tags and results exclude punctuation, while the results for all other languages use gold part-of-speech tags and include punctuation.

Model	Du	Cz	Pt	Da	Sw
	Prague				
Proj GSib	80.45	85.12	88.85	88.17	85.50
Proj Edge	80.38	84.04	88.14	88.29	86.09
1-EC CS-GSib	82.78	85.90	89.74	88.64	85.70
1-EC Edge	83.33	84.97	89.21	88.19	86.46
	Mel’čukian				
Proj GSib	82.26	87.96	89.19	90.23	89.59
Proj Edge	82.09	86.18	88.73	89.29	89.00
1-EC CS-GSib	86.03	87.89	90.34	90.50	89.34
1-EC Edge	85.28	87.57	89.96	90.14	88.97
	Stanford				
Proj GSib	81.16	86.83	88.80	88.84	87.27
Proj Edge	80.56	86.18	88.61	88.69	87.92
1-EC CS-GSib	84.67	88.34	90.20	89.22	88.15
1-EC Edge	83.62	87.13	89.43	88.74	87.36

Table 4: Overall Unlabeled Attachment Scores (UAS) for all words.³ CS-GSib is the proposed crossing-sensitive grandparent-sibling factorization. For each data set, we bold the most accurate model and those not significantly different from the most accurate (sign test, $p < .05$). Languages are sorted in increasing order of projectivity.

Model	UAS
Proj GSib	93.10
Proj Edge	92.63
1-EC CS-GSib	93.22
1-EC Edge	92.80

Table 5: English results

the parser is able to score with a sibling context more often than it is able to score with a grandparent, perhaps explaining why the datasets using the Stanford conjunction representation saw the largest gains from including the higher order factors into the 1-Endpoint-Crossing parser.

Across languages, the third-order 1-Endpoint-Crossing parser runs 2.1-2.7 times slower than the third-order projective parser (71-104 words per second, compared with 183-268 words per second). Parsing speed is correlated with the amount of pruning. The level of pruning mentioned earlier is relatively permissive, retaining 39.0-60.7% of the edges in the complete graph; a higher level of pruning could likely achieve much faster parsing times with the same underlying parsing algorithms.

Part Used	Du	Cz	Pt	Da	Sw
	Mel'čukian				
CrossedEdge	8.5	4.5	3.2	1.4	1.2
GrandSib	81.2	89.1	90.7	95.7	96.2
Grand	1.1	0.5	0.8	0.3	0.2
Sib	9.0	5.8	5.2	2.6	2.3
Edge	< 0.1	< 0.1	0	< 0.1	0
	Stanford				
CrossedEdge	8.4	5.1	3.3	2.0	1.8
GrandSib	81.4	87.8	90.5	94.2	95.2
Grand	1.1	0.5	0.7	0.3	0.3
Sib	8.9	6.5	5.2	3.5	2.6
Edge	< 0.1	0.1	0	< 0.1	0

Table 6: The proportion of edges in the predicted output trees from the CS-GSib 1-Endpoint-Crossing parser that would have used each of the five part types for scoring.

6 Discussion

There have been many other notable approaches to non-projective parsing with larger scopes than single edges, including transition-based parsers, directed spanning tree graph-based parsers, and mildly non-projective graph-based parsers.

Transition-based parsers score actions that the parser may take to transition between different configurations. These parsers typically use either greedy or beam search, and can condition on any tree context that is in the history of the parser's actions so far. Zhang and Nivre (2011) significantly improved the accuracy of an arc-eager transition system (Nivre, 2003) by adding several additional classes of features, including some third-order features. Basic arc-eager and arc-standard (Nivre, 2004) models that parse left-to-right using a stack produce projective trees, but transition-based parsers can be modified to produce crossing edges. Such modifications include pseudo-projective parsing in which the dependency labels encode transformations to be applied to the tree (Nivre and Nilsson, 2005), adding actions that add edges to words in the stack that are not the topmost item (Attardi, 2006), adding actions that swap the positions of words (Nivre, 2009), and adding a second stack (Gómez-Rodríguez and Nivre, 2010).

Graph-based approaches to non-projective parsing either consider all directed spanning trees or restricted classes of mildly non-projective trees. Directed spanning tree approaches with higher order features either use approximate learning techniques,

such as loopy belief propagation (Smith and Eisner, 2008), or use dual decomposition to solve relaxations of the problem (Koo et al., 2010; Martins et al., 2013). While not guaranteed to produce optimal trees within a fixed number of iterations, these dual decomposition techniques do give certificates of optimality on the instances in which the relaxation is tight and the algorithm converges quickly.

This paper described a mildly non-projective graph-based parser. Other parsers in this class find the optimal tree in the class of well-nested, block degree two trees (Gómez-Rodríguez et al., 2011), or in a class of trees further restricted based on gap inheritance (Pitler et al., 2012) or the head-split property (Satta and Kuhlmann, 2013), with edge-factored running times of $O(n^5) - O(n^7)$. The factorization used in this paper is not immediately compatible with these parsers: the complex cases in these parsers are due to *gaps*, not *crossings*. However, there may be analogous “gap-sensitive” factorizations that could allow these parsers to be extended without large increases in running times.

7 Conclusion

This paper proposed an exact, graph-based algorithm for non-projective parsing with higher order features. The resulting parser has the same asymptotic run time as a third-order projective parser, and is significantly more accurate for many experimental settings. An exploration of other factorizations that facilitate non-projective parsing (for example, an analogous “gap-sensitive” variant) may be an interesting avenue for future work. Recent work has investigated faster variants for third-order graph-based projective parsing (Rush and Petrov, 2012; Zhang and McDonald, 2012) using structured prediction cascades (Weiss and Taskar, 2010) and cube pruning (Chiang, 2007). It would be interesting to extend these lines of work to the crossing-sensitive third-order parser as well.

Acknowledgments

I would like to thank Sampath Kannan, Mitch Marcus, Chris Callison-Burch, Michael Collins, Mark Liberman, Ben Taskar, Joakim Nivre, and the three anonymous reviewers for valuable comments on earlier versions of this material.

References

- G. Attardi. 2006. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of CoNLL*, pages 166–170.
- A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2001. The Prague Dependency Treebank: Three-level annotation scenario. In Anne Abeillé, editor, *Treebanks: Building and Using Syntactically Annotated Corpora*, pages 103–127. Kluwer Academic Publishers.
- B. Bohnet and J. Kuhn. 2012. The best of both worlds – a graph-based completion model for transition-based parsers. In *Proceedings of EACL*, pages 77–87.
- S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL*, pages 149–164.
- X. Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, pages 957–961.
- E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of ACL*, pages 173–180.
- D. Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.
- M. De Marneffe and C. Manning. 2008. Stanford typed dependencies manual.
- J. Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In Harry Bunt and Anton Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62. Kluwer Academic Publishers.
- C. Gómez-Rodríguez and J. Nivre. 2010. A transition-based parser for 2-planar dependency structures. In *Proceedings of ACL*, pages 1492–1501.
- C. Gómez-Rodríguez, J. Carroll, and D. Weir. 2011. Dependency parsing schemata and mildly non-projective dependency parsing. *Computational Linguistics*, 37(3):541–586.
- K. Hall. 2007. K-best spanning tree parsing. In *Proceedings of ACL*, pages 392–399.
- R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proceedings of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*, pages 105–112.
- T. Koo and M. Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL*, pages 1–11.
- T. Koo, A. M. Rush, M. Collins, T. Jaakkola, and D. Sonntag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of EMNLP*, pages 1288–1298.
- T. Koo. 2010. *Advances in discriminative dependency parsing*. Ph.D. thesis, Massachusetts Institute of Technology.
- S. Kübler, R. McDonald, and J. Nivre. 2009. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 2(1):1–127.
- M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.
- A. F. T. Martins, N. A. Smith, and E. P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of ACL*, pages 342–350.
- A. Martins, M. Almeida, and N. A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of ACL (Short Papers)*, pages 617–622.
- R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, pages 81–88.
- R. McDonald and G. Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of the 10th International Conference on Parsing Technologies*, pages 121–132.
- R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT/EMNLP*, pages 523–530.
- I. Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press.
- J. Nivre and J. Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of ACL*, pages 99–106.
- 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*, pages 915–932.
- J. Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies*, pages 149–160.
- J. Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57.
- J. Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of ACL*, pages 351–359.
- E. Pitler, S. Kannan, and M. Marcus. 2012. Dynamic programming for higher order parsing of gap-minding trees. In *Proceedings of EMNLP*, pages 478–488.
- E. Pitler, S. Kannan, and M. Marcus. 2013. Finding optimal 1-Endpoint-Crossing trees. *Transactions of the Association for Computational Linguistics*, 1(Mar):13–24.

- A. Rush and S. Petrov. 2012. Vine pruning for efficient multi-pass dependency parsing. In *Proceedings of NAACL*, pages 498–507.
- G. Satta and M. Kuhlmann. 2013. Efficient parsing for head-split dependency trees. *Transactions of the Association for Computational Linguistics*, 1(July):267–278.
- D. A. Smith and J. Eisner. 2008. Dependency parsing by belief propagation. In *Proceedings of EMNLP*, pages 145–156.
- D. Weiss and B. Taskar. 2010. Structured Prediction Cascades. In *AISTATS*, pages 916–923.
- D. Zeman, D. Mareček, M. Popel, L. Ramasamy, J. Štěpánek, Z. Žabokrtský, and J. Hajič. 2012. HamleDT: To parse or not to parse? In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, pages 2735–2741.
- H. Zhang and R. McDonald. 2012. Generalized higher-order dependency parsing with cube pruning. In *Proceedings of EMNLP*, pages 320–331.
- Y. Zhang and J. Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of ACL (Short Papers)*, pages 188–193.

Exploring the Role of Stress in Bayesian Word Segmentation using Adaptor Grammars

Benjamin Börschinger^{1,2} Mark Johnson^{1,3}

¹Department of Computing, Macquarie University, Sydney, Australia

²Department of Computational Linguistics, Heidelberg University, Heidelberg, Germany

³Santa Fe Institute, Santa Fe, USA

{benjamin.borschinger|mark.johnson}@mq.edu.au

Abstract

Stress has long been established as a major cue in word segmentation for English infants. We show that enabling a current state-of-the-art Bayesian word segmentation model to take advantage of stress cues noticeably improves its performance. We find that the improvements range from 10 to 4%, depending on both the use of phonotactic cues and, to a lesser extent, the amount of evidence available to the learner. We also find that in particular early on, stress cues are much more useful for our model than phonotactic cues by themselves, consistent with the finding that children do seem to use stress cues before they use phonotactic cues. Finally, we study how the model's knowledge about stress patterns evolves over time. We not only find that our model correctly acquires the most frequent patterns relatively quickly but also that the Unique Stress Constraint that is at the heart of a previously proposed model does not need to be built in but can be acquired jointly with word segmentation.

1 Introduction

Among the first tasks a child language learner has to solve is picking out words from the fluent speech that constitutes its linguistic input.¹ For English, stress has long been claimed to be a useful cue in infant word segmentation (Jusczyk et al., 1993; Jusczyk et al., 1999b), following the demonstra-

tion of its effectiveness in adult speech processing (Cutler et al., 1986). Several studies have investigated the role of stress in word segmentation using computational models, using both neural network and “algebraic” (as opposed to “statistical”) approaches (Christiansen et al., 1998; Yang, 2004; Lignos and Yang, 2010; Lignos, 2011; Lignos, 2012). Bayesian models of word segmentation (Brent, 1999; Goldwater, 2007), however, have until recently completely ignored stress. The sole exception in this respect is Doyle and Levy (2013) who added stress cues to the Bigram model (Goldwater et al., 2009), demonstrating that this leads to an improvement in segmentation performance. In this paper, we extend their work and show how to integrate stress cues into the flexible Adaptor Grammar framework (Johnson et al., 2007). This allows us to both start from a stronger baseline model and to investigate how the role of stress cues interacts with other aspects of the model. In particular, we find that phonotactic cues to word-boundaries interact with stress cues, indicating synergistic effects for small inputs and partial redundancy for larger inputs. Overall, we find that stress cues add roughly 6% token f-score to a model that does not account for phonotactics and 4% to a model that already incorporates phonotactics. Relatedly and in line with the finding that stress cues are used by infants before phonotactic cues (Jusczyk et al., 1999a), we observe that phonotactic cues require more input than stress cues to be used efficiently. A closer look at the knowledge acquired by our models shows that the Unique Stress Constraint of Yang (2004) can be acquired jointly with segmenting the input instead

¹The datasets and software to replicate our experiments are available from <http://web.science.mq.edu.au/~bborschi/>

of having to be pre-specified; and that our models correctly identify the predominant stress pattern of the input but underestimate the frequency of iambic words, which have been found to be missegmented by infant learners.

The outline of the paper is as follows. In Section 2 we review prior work. In Section 3 we introduce our own models. In Section 4 we explain our experimental evaluation and its results. Section 5 discusses our findings, and Section 6 concludes and provides some suggestions for future research.

2 Background and related work

Lexical stress is the “accentuation of syllables within words” (Cutler, 2005) and has long been argued to play an important role in adult word recognition. Following Cutler and Carter (1987)’s observation that stressed syllables tend to occur at the beginnings of words in English, Jusczyk et al. (1993) investigated whether infants acquiring English take advantage of this fact. Their study demonstrated that this is indeed the case for 9 month olds, although they found no indication of using stressed syllables as cues for word boundaries in 6 month olds. Their findings have been replicated and extended in subsequent work (Jusczyk et al., 1999b; Thiessen and Saffran, 2003; Curtin et al., 2005; Thiessen and Saffran, 2007). A brief summary of the key findings is as follows: English infants treat stressed syllables as cues for the beginnings of words from roughly 7 months of age, suggesting that the role played by stress needs to be acquired, and that this requires antecedent segmentation by non-stress-based means (Thiessen and Saffran, 2007). They also exhibit a preference for low-pass filtered stress-initial words from this age, suggesting that it is indeed stress and not other phonetic or phonotactic properties that are treated as a cue for word-beginnings (Jusczyk et al., 1993). In fact, phonotactic cues seem to be used later than stress cues (Jusczyk et al., 1999a) and seem to be outweighed by stress cues (Mattys and Jusczyk, 2000).

The earliest computational model for word segmentation incorporating stress cues we are aware of is the recurrent network model of Christiansen et al. (1998) and Christiansen and Curtin (1999). They only reported a word-token f-score of 44% (roughly, segmentation accuracy: see Section 4), which is

considerably below the performance of subsequent models, making a direct comparison complicated. Yang (2004) introduced a simple incremental algorithm that relies on stress by embodying a Unique Stress Constraint (USC) that allows at most a single stressed syllable per word. On pre-syllabified child directed speech, he reported a word token f-score of 85.6% for a non-statistical algorithm that exploits the USC. While the USC has been argued to be near-to-universal and follows from the “culminative function of stress” (Fromkin, 2001; Cutler, 2005), the high score Yang reported crucially depends on every word token carrying stress, including function words. More recently, Lignos (2010, 2011, 2012) further explored Yang’s original algorithm, taking into account that function words should not be assumed to possess lexical stress cues. While his scores are in line with those reported by Yang, the importance of stress for this learner were more modest, providing a gain of around 2.5% (Lignos, 2011). Also, the Yang/Lignos learner is unable to acquire knowledge about the role stress plays in the language, e.g. that stress tends to fall on particular positions within words.

Doyle and Levy (2013) extend the Bigram model of Goldwater et al. (2009) by adding stress-templates to the lexical generator. A stress-template indicates how many syllables the word has, and which of these syllables (if any) are stressed. This allows the model to acquire knowledge about the stress patterns of its input by assigning different probabilities to the different stress-templates. However, Doyle and Levy (2013) do not directly examine the probabilities assigned to the stress-templates; they only report that their model does slightly prefer stress-initial words over the baseline model by calculating the fraction of stress-initial word types in the output segmentations of their models. They also demonstrate that stress cues do indeed aid segmentation, although their reported gain of 1% in token f-score is even smaller than that reported by Lignos (2011). Our own approach differs from theirs in assuming phonemic rather than pre-syllabified input (although our model could, trivially, be run on syllabified input as well) and makes use of Adaptor Grammars instead of the Goldwater et al. (2009) Bigram model, providing us with a flexible framework for exploring the usefulness of stress in differ-

ent models.

Adaptor Grammar (Johnson et al., 2007) is a grammar-based formalism for specifying non-parametric hierarchical models. Previous work explored the usefulness of, for example, syllable-structure (Johnson, 2008b; Johnson and Goldwater, 2009) or morphology (Johnson, 2008b; Johnson, 2008a) in word segmentation. The closest work to our own is Johnson and Demuth (2010) who investigate the usefulness of tones for Mandarin phonemic segmentation. Their way of adding tones to a model of word segmentation is very similar to our way of incorporating stress.

3 Models

We give an intuitive description of the mathematical background of Adaptor Grammars in 3.1, referring the reader to Johnson et al. (2007) for technical details. The models we examine are derived from the collocational model of Johnson and Goldwater (2009) by varying three parameters, resulting in 6 models: two baselines that do not take advantage of stress cues and either do or do not use phonotactics, as described in Section 3.2; and four stress models that differ with respect to the use of phonotactics, and as to whether they embody the Unique Stress Constraint introduced by Yang (2004). We describe these models in section 3.3.

3.1 Adaptor Grammars

Briefly, an Adaptor Grammar (AG) can be seen as a probabilistic context-free grammar (PCFG) with a special set of *adapted* non-terminals. We use underlining to distinguish adapted non-terminals (\underline{X}) from non-adapted non-terminals (Y). The distribution for each adapted non-terminal \underline{X} is drawn from a Pitman-Yor Process which takes as its base-distribution the tree-distribution over trees rooted in \underline{X} as defined by the PCFG. As an effect, each adapted non-terminal can be seen as having associated with it a cache of previously-generated subtrees that can be reused without having to be regenerated using the individual PCFG rules. This allows AGs to learn reusable sub-trees such as words, sequences of words, or smaller units such as Onsets and Codas. Thus, while ordinary PCFGs have a finite number of parameters (one probability for each rule), Adaptor Grammars in addition have a parameter for every

possible complete tree rooted in any of its adapted non-terminals, leading to a potentially infinite number of such parameters. The Pitman-Yor Process induces a rich-get-richer dynamics, biasing the model towards identifying a small set of units that can be reused as often as possible. In the case of word segmentation, the model will try to identify as compact a lexicon as possible to segment the unsegmented input.

3.2 Baseline models

Our starting point is the state-of-the-art AG model for word segmentation, Johnson and Goldwater (2009)’s colloc3-syll model, reproduced in Figure 1.² The model assumes that words are grouped into larger collocational units that themselves can be grouped into even larger collocational units. This accounts for the fact that in natural language, there are strong word-to-word dependencies that need to be accounted for if severe undersegmentations of the form “is in the” are to be avoided (Goldwater, 2007; Johnson and Goldwater, 2009; Börschinger et al., 2012). It also uses a language-independent form of syllable structure to constrain the space of possible words. Finally, this model can learn word-initial onsets and word-final codas. In a language like English, this ability provides additional cues to word-boundaries as certain onsets are much more likely to occur word-initially than medially (e.g. “bl” in “black”), and analogously for certain codas (e.g. “dth” in “width” or “ngth” in “strength”).

We define an additional baseline model by replacing rules (5) and (6) by (17), and deleting rules (7) to (12). This removes the model’s ability to use phonotactic cues to word-boundaries.

$$\underline{\text{Word}} \rightarrow \text{Syll}(\text{Syll})(\text{Syll})(\text{Syll}) \quad (17)$$

We refer to the model in Figure 1 as the colloc3-phon model, and the model that results from substituting and removing rules as described as the colloc3-nophon model. Alternatively, one could limit the models ability to capture word-to-word dependencies by removing rules (1) to (3). This results

²We follow Johnson and Goldwater (2009) in limiting the length of possible words to four syllables to speed up runtime. In pilot experiments, this choice did not have a noticeable effect on segmentation performance.

<u>Collocation3</u> → <u>Collocation3</u> ⁺	(1)	<u>Word</u> → { <u>SSyll</u> <u>USyll</u> } ^{1,4}	(18)
<u>Collocation3</u> → <u>Collocation2</u> ⁺	(2)	<u>SSyll</u> → (<u>Onset</u>) RhymeS	(19)
<u>Collocation2</u> → <u>Collocation</u> ⁺	(3)	<u>USyll</u> → (<u>Onset</u>) RhymeU	(20)
<u>Collocation</u> → <u>Word</u> ⁺	(4)	RhymeS → Vowel * (<u>Coda</u>)	(21)
<u>Word</u> → SyllIF	(5)	RhymeU → Vowel (<u>Coda</u>)	(22)
<u>Word</u> → SyllI (Syll) (Syll) SyllF	(6)	<u>Onset</u> → Consonant ⁺	(23)
SyllIF → (<u>OnsetI</u>) RhymeF	(7)	<u>Coda</u> → Consonant ⁺	(24)
SyllI → (<u>OnsetI</u>) Rhyme	(8)		
SyllF → (<u>Onset</u>) RhymeF	(9)		
<u>CodaF</u> → Consonant ⁺	(10)		
RhymeF → Vowel (<u>CodaF</u>)	(11)		
<u>OnsetI</u> → Consonant ⁺	(12)		
Syll → (<u>Onset</u>) Rhyme	(13)		
Rhyme → Vowel (<u>Coda</u>)	(14)		
<u>Onset</u> → Consonant ⁺	(15)		
<u>Coda</u> → Consonant ⁺	(16)		

Figure 2: Description of the all-stress-patterns model. We use $X^{\{m,n\}}$ for “at least m and at most n repetitions of X ” and $\{X | Y\}$ for “either X or Y ”. Stress is associated with a vowel by suffixing it with the special terminal symbol $*$, leading to a distinction between stressed ($SSyll$) and unstressed ($USyll$) syllables. A word can consist of any possible sequence of up to four syllables, as indicated by the regular-expression notation. By additionally adding initial and final variants of $SSyll$ and $USyll$ as in Figure 1, phonotactics can be combined with stress cues.

Figure 1: The baseline model. We use regular-expression notation to abbreviate multiple rules. $X^{\{n\}}$ stands for up to n repetitions of X , brackets indicate optionality, and X^+ stands for one or more repetitions of X . \underline{X} indicates an adapted non-terminal. Rules that introduce terminals for the pre-terminals $Vowel$, $Consonant$ are omitted. Refer to the main text for an explanation of the grammar.

in the colloc-model (Johnson, 2008b) that has previously been found to behave similarly to the Bigram model used in Doyle and Levy (2013) (Johnson, 2008b; Börschinger et al., 2012). We performed experiments with the colloc-model as well and found similar results to Doyle and Levy (2013) which are, while overall worse, similar in trend to the results obtained for the colloc3-models. For the rest of the paper, therefore, we will focus on variants of the colloc3-model.

3.3 Stress-based models

In order for stress cues to be helpful, the model must have some way of associating the position of stress with word-boundaries. Intuitively, the reason stress helps infants in segmenting English is that a stressed syllable is a reliable indicator of the beginning of a word (Jusczyk et al., 1993). More generally, if there is a (reasonably) reliable relationship between the position of stressed syllables and beginnings (or

endings) of words, a learner might exploit this relationship. In a Bayesian model, this intuition can be captured by modifying the lexical generator, that is, the distribution that generates Words.

Here, changing the lexical generator corresponds to modifying the rules expanding Word. A straightforward way to modify it accordingly is to enumerate all possible sequences of stressed and unstressed syllables.³ A lexical generator like this is given in Figure 2. In the data, stress cues are represented using a special terminal “ $*$ ” that follows a stressed vowel, as illustrated in Figure 3. In the grammar, “ $*$ ” is constrained to only surface following a $Vowel$, rendering a syllable in which it occurs stressed ($SSyll$). Syllables that do not contain a “ $*$ ” are considered unstressed ($USyll$). By performing inference for the probabilities assigned to the different expansions of rule (18), our models can, for example, learn that a bi-syllabic word that is stress-initial (a trochee) is more probable than one that puts stress on the second syllable (an iamb). This (partly) captures the tendency of English for stress-initial words and thus provide an additional cue for identifying words; and it is exactly the kind of preference infant learners of English seem to acquire (Jusczyk

³This is, in essence, also the strategy chosen by Doyle and Levy (2013).

grammar	phon	stress	USC
colloc3-nophon			
colloc3-phon	•		
colloc3-nophon-stress		•	
colloc3-phon-stress	•	•	
colloc3-nophon-stress-usc		•	•
colloc3-phon-stress-usc	•	•	•

Table 1: The different models used in our experiments. “phon” indicates whether phonotactics are used, “stress” whether stress cues are used and “usc” whether the Unique Stress Constraint is assumed.

orthographic	the do -gie
no-stress	dh ah d ao g iy
stress	dh ah d ao * g iy

Figure 3: Illustration of the input-representation we choose. We indicate primary stress according to the dictionary with bold-face in the orthography. The phonemic transcription uses ARPABET and is produced using an extended version of CMUDict. Primary stress is indicated by inserting the special symbol “*” after the vowel of a stressed syllable.

et al., 1993).

We can combine this lexical generator with the colloc3-nophon baseline, resulting in the colloc3-nophon-stress model. We can also add phonotactics to the lexical generator in Figure 2 by adding initial and final variants of SSyll and USyll, analogous to rules (5) to (12) in Figure 1. This yields the colloc3-phon-stress model. We can also add the Unique Stress Constraint (USC) (Yang, 2004) by excluding all variants of rule (18) that generate two or more stressed syllables. For example, while the lexical generator for the colloc3-nophon-stress model will include the rule $\underline{\text{Word}} \rightarrow \text{SSyll SSyll}$, the lexical generator embodying the USC lacks this rule. We refer to the models that include the USC as colloc3-nophon-stress-usc and colloc3-phon-stress-usc models. A compact overview of the six different models is given in Table 1.

4 Experiments

We evaluate our models on several corpora of child directed speech. We first describe the corpora we used, then the experimental methodology employed and finally the experimental results. As the trend is

comparable across all corpora, we only discuss in detail results obtained on the Alex corpus. For completeness, however, Table 3 reports the “standard” evaluation of performing inference over all of the three corpora.

4.1 Corpora and corpus creation

Following Christiansen et al. (1998) and Doyle and Levy (2013), we use the Korman corpus (Korman, 1984) as one of our corpora. It comprises child-directed speech for very young infants, aged between 6 and 16 weeks and, like all other corpora used in this paper, is available through the CHILDES database (MacWhinney, 2000). We derive a phonemicized version of the corpus using an extended version of CMUDict (Carnegie Mellon University, 2008)⁴, as we were unable to obtain the stress-annotated version of this corpus used in previous experiments. The phonemicized version is produced by replacing each orthographic word in the transcript with the first pronunciation given by the dictionary. CMUDict also annotates lexical stress, and we use this information to add stress cues to the corpus. We only code primary lexical stresses in the input, ignoring secondary stresses in line with experimental work that indicates that human listeners are capable of reliably distinguishing primary and secondary stress (Mattys, 2000). Due to the very low frequency of words with 3 or more syllables in these corpora, this choice has very little effect on the number of stress cues available in the input. Our version of the Korman corpus contains, in total, 11413 utterances. Unlike Christiansen et al. (1998), Yang (2004), and Doyle and Levy (2013), we follow Lignos and Yang (2010) in making the more realistic assumption that the 94 mono-syllabic function words listed by Selkirk (1984) never surface with lexical stress. As function words account for roughly 50% of the tokens but only roughly 5% of the types in our corpora, this means that the type and token distribution of stress patterns differs dramatically in all our corpora, as can be seen from Table 2.

We also added stress information to the Brent-Bernstein-Ratner corpus (Bernstein-Ratner, 1987; Brent, 1999), following the procedure just outlined. This corpus is a de-facto standard for evaluat-

⁴<http://svn.code.sf.net/p/cmuspinx/code/trunk/cmudict/cmudict.0.7a>

Pattern	brent		korman		alex	
	Tok	Typ	Tok	Typ	Tok	Typ
W ⁺	.48	.07	.47	.08	.44	.05
SW [*]	.49	.86	.49	.86	.52	.87
WSW [*]	.03	.07	.03	.06	.04	.07
Other	.00	.00	.00	.00	.00	.00

Table 2: Relative frequencies for stress patterns for the corpora used in our study. X^* stands for 0 or more, X^+ for one or more repetitions of X , and S for a stressed and W for an unstressed syllable. Note the stark asymmetry between type and token frequencies for unstressed words. Up to two-decimal places, patterns other than the ones given have relative frequency 0.00 (frequencies might not sum to 1 as an artefact of rounding to 2 decimal places).

ing models of Bayesian word segmentation (Brent, 1999; Goldwater, 2007; Goldwater et al., 2009; Johnson and Goldwater, 2009), comprising in total 9790 utterances.

As our third corpus, we use the Alex portion of the Providence corpus (Demuth et al., 2006; Börschinger et al., 2012). A major benefit of the Providence corpus is that the video-recordings from which the transcripts were produced are available through CHILDES alongside the transcripts. This will allow future work to rely on even more realistic stress cues that can be derived directly from the acoustic signal. While beyond the scope of this paper, we believe choosing a corpus that makes richer information available will be important for future work on stress (and other acoustic) cues. Another major benefit of the Alex corpus is that it provides longitudinal data for a single infant, rather than being a concatenation of transcripts collected from multiple children, such as the Korman and the Brent-Bernstein-Ratner corpus. In total, the Alex corpus comprises 17948 utterances.

Note that despite the differences in age of the infants and overall make-up of the corpora, the distribution of stress patterns across the corpora is roughly the same, as shown by Table 2 for the first 10,000 utterances of each of the corpora. This suggests that the distribution of stress patterns both at a token and type level is a robust property of English child-directed speech.

4.2 Evaluation procedure

The aim of our experiments is to understand the contribution of stress cues to the Bayesian word segmentation models described in Section 3. To get an idea of how input size interacts with this, we look at prefixes of the corpora with increasing sizes (100, 200, 500, 1000, 2000, 5000, and 10,000 utterances). In addition, we are interested in understanding what kind of stress pattern preferences our models acquire. For this, we also collect samples of the probabilities assigned to the different expansions of rule (18), allowing us to examine this directly. The standard evaluation of segmentation models involves having them segment their input in an unsupervised manner and evaluating performance on how well they segmented that input. We additionally evaluate the models on a test set for each corpus. Use of a separate test set has previously been suggested as a means of testing how well the knowledge a learner acquired generalizes to novel utterances (Pearl et al., 2011), and is required for the kind of comparison across different sizes of input we are interested in to determine whether the role of stress cues interacts with the input size.

We create the test-sets by taking the final 1000 utterances for each corpus. These 1000 utterances will be segmented by the model after it has performed inference on its input, without making any further changes to the lexicon that the model has induced. In other words, the model will have to segment each of the test utterances using only the lexicon (and any additional knowledge about co-occurrences, phonotactics, and stress) it has acquired from the training portion of the corpus during inference.

We measure segmentation performance using the standard metric of token f-score (Brent, 1999) which is the harmonic mean of token precision and recall. Token f-score provides an overall impression of how accurate individual word tokens were identified. To illustrate, if the gold segmentation is “the dog”, the segmentation “th e dog” has a token precision of $\frac{1}{3}$ (one out of three predicted words is correct); a token recall of $\frac{1}{2}$ (one of the two gold words was correctly identified); and a token f-score of 0.4.

4.3 Inference

For inference, we closely follow Johnson and Goldwater (2009): we put vague priors on all the hyper-

p	s	usc	alex		korman		brent	
			train	test	train	test	train	test
•			.81	.81	.85	.83	.82	.82
			.85	.84	.86	.84	.86	.86
•	•	•	.86	.87	.87	.86	.86	.87
			.88	.88	.88	.87	.87	.87
			.87	.88	.87	.88	.86	.87
•	•	•	.88	.88	.88	.87	.87	.88

Table 3: Token f-scores on both train and test portions for all three corpora when inference is performed over the full corpus. Note that the benefit of stress is clearer when evaluating on the test set, and that overall, performance of the different models is comparable across all three corpora. Models are coded according to the key in Table 1.

parameters of our models and run 4 chains for 1000 iterations, collecting 20 samples from each chain with a lag of 10 iterations between each sample after a burn-in of 800 iterations, using both batch-initialization and table-label resampling to ensure good convergence of the sampler. We construct a single segmentation from the posterior samples using their minimum Bayes risk decoding, providing a single score for each condition.

4.4 Experimental conditions

Each of our six models is evaluated on inputs of increasing size, starting at 100 and ending at 10,000 utterances, allowing us to investigate both how performance and “knowledge” of the learner varies as a function of input size. For completeness, we also report the “standard” evaluation, i.e. performance of our models on all corpora when trained on the entire input in Table 3. We will focus our discussion on the results obtained on the Alex corpus, which are depicted in Figure 4, where the input size is depicted on the x-axis, and the segmentation f-score for the test-set on the y-axis.

5 Discussion

We find a clear improvement for the stress-models over both the colloc3-nophon and the colloc3-phon models. As can be seen in Table 3, the overall trend is the same for all three corpora, both when evaluating on the input and the separate test-set.⁵

⁵We performed Wilcoxon rank sum tests on the individual scores of the 4 independent chains for each model on the full training data sets and found that the stress-models were always

Note how the relative gain for stress is roughly 1% higher when evaluating on the test-set; this might have to do with Jusczyk (1997)’s observation that the advantage of stress “might be more evident for relatively unexpected or unfamiliarized strings” (Jusczyk, 1997). A closer look at Figure 4 indicates further interesting differences between the colloc3-nophon and the colloc3-phon models that only become evident when considering different input sizes.

5.1 Stress cues without phonotactics

For the colloc3-nophon models, we observe a relatively stable improvement by adding stress cues of 6-7%, irrespective of input size and whether or not the Unique Stress Constraint (USC) is assumed. The sole exception to this occurs when the learner only gets to see 100 utterances: in this case, the colloc-nophon-stress model only shows a 3% improvement, whereas the colloc3-nophon-stress-usc model obtains a boost of roughly 8%. Noticeable consistent differences between the colloc3-nophon-stress and colloc3-nophon-stress-usc model, however, all but disappear starting from around 500 utterances. This is somewhat surprising, considering that it is the USC that was argued by Yang (2004) to be key for taking advantage of stress.⁶

We take this behaviour to indicate that even with as little evidence as 200 to 500 utterances, a Bayesian ideal learner can effectively infer that something like the USC is true of English. This also becomes clear when examining how the learners’ preferences for different stress patterns evolve over time, as we do in Section 5.3 below.

5.2 Stress cues and phonotactics

Overall, the models including phonotactic cues perform better than those that do not rely on phonotactics. However, the overall gain contributed by stress to the colloc3-phon baseline is smaller, al-

significantly more accurate ($p < 0.05$) than the baseline models except when evaluating on the training data for the Korman and Brent corpora.

⁶On data in which function words are marked for stress (as in Yang (2004) and Doyle and Levy (2013)), the USC yields extremely high scores across all models, simply because roughly every second word is a function word. Given that this assumption is extremely unnatural, we do not take this as an argument for the USC.

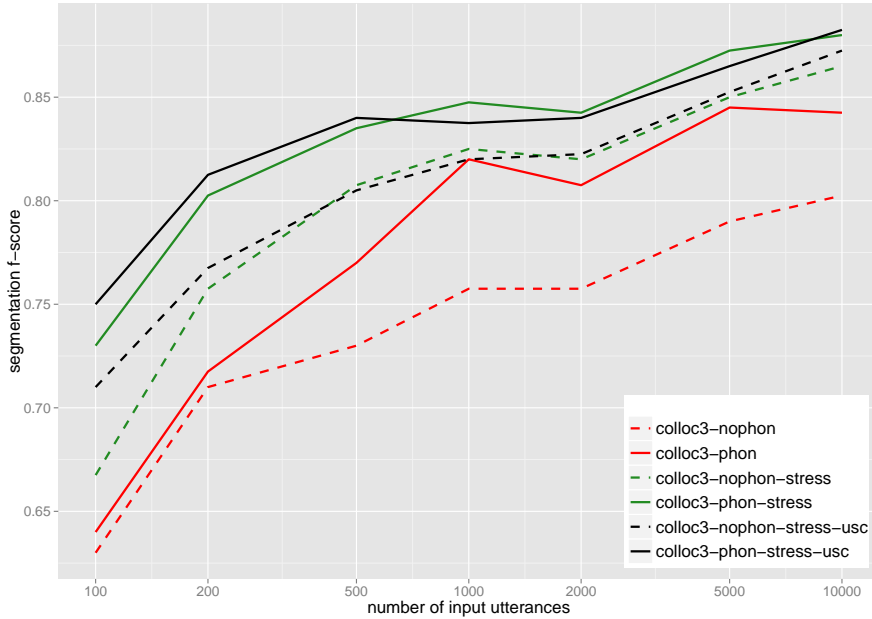


Figure 4: Segmentation performance of the different models, across different input sizes and as evaluated on the test-set for the Alex corpus. The no-stress baselines are given in red, the stress-models without the Unique Stress Constraint (USC) in green and the ones including the USC in black. Solid lines indicate models that use, dashed lines models that do not use phonotactics. Refer to the text for discussion.

though this seems to depend on the size of the input. While phonotactics by itself appears to be a powerful cue, yielding a noticeable 4-5% improvement over the colloc3-nophon baseline, the learner seems to require at least around 500 utterances before the colloc3-phon model becomes clearly more accurate than the colloc3-nophon model. In contrast, even for only 100 utterances stress cues by themselves provide a 3% improvement to the colloc3-nophon model, indicating that they can be taken advantage of earlier. While the number of utterances processed by a Bayesian ideal learner is not directly related to developmental stages, this observation is consistent with the psycholinguists' claim that phonotactics are used by infants for word segmentation after they have begun to use stress for segmentation (Jusczyk et al., 1999a).

Turning to the interaction between stress and phonotactics, we see that there is no consistent advantage of including the USC in the model. This is, in fact, even clearer than for the colloc3-nophon model where at least for small inputs of size 100, the USC added almost 5% in performance. For the colloc3-phon models, we only observe a 1-2% improvement by adding the USC up until 500 utter-

ances. This further strengthens the point that even in the absence of such an innate constraint, a statistical learner can take advantage of stress cues and, as we show below, actually acquire something like the USC from the input.

The 4% difference between the colloc3-phon-stress / colloc3-phon-stress-usc models to the colloc3-phon baseline is smaller than the 7% difference between the colloc3-nophon and colloc3-nophon-stress models. This shows that there is a redundancy between phonotactic and stress cues in large amounts of data, as their joint contribution to the colloc3-nophon baseline is less than the sum of their individual contributions at 10,000 utterances, of 4% (for phonotactics) and 7% (for stress).

Unlike for the colloc3-nophon models, we also see a clear impact of input size. In particular, at 100 utterances the addition of stress cues leads to an 8 – 10% improvement, depending on whether or not the USC is assumed, whereas for the colloc3-nophon model we only observed a 3 – 8% improvement. This is particularly striking when we consider that by themselves, the phonotactic cues only contribute a 1% improvement to the colloc3-nophon baseline when trained on the 100 utterance corpus,

indicating a synergistic interaction (rather than redundancy) between phonotactics and stress for small inputs. This effect disappears starting from around 1000 utterances; for inputs of size 1000 and larger, the net-gain of stress drops from roughly 10% to a 3–4% improvement. That is, while we did not notice any relationship between input size and impact of stress cues for the colloc3-nophon model, we do see such an interaction for the combination of phonotactics and stress cues which, taken together, lead to a larger relative gain in performance on smaller inputs than on large ones.

5.3 Acquisition of stress patterns

In addition to acquiring a lexicon, the Bayesian learner acquires knowledge about the possible stress patterns of English words. The fact that this knowledge is explicitly represented through the PCFG rules and their probabilities that define the lexical generator allows us to study the generalisations about stress the model actually acquires. While Doyle and Levy (2013) suggest carrying out such an analysis, they restrict themselves to estimating the fraction of stress patterns in the segmented output. As shown in Table 2, however, the type and token distributions of stress patterns can differ substantially. We therefore investigate the stress preferences acquired by our learner by examining the probabilities assigned to the different expansions of rule (18), aggregating the probabilities of the individual rules into patterns. For example, the rules $\underline{\text{Word}} \rightarrow \text{SSyll}(\text{USyll})^{\{0,3\}}$ correspond to the pattern “Stress on the first syllable”, whereas the rules $\underline{\text{Word}} \rightarrow \text{USyll}^{\{1,4\}}$ correspond to the pattern “Unstressed word”. By computing the respective probabilities, we get the overall probability assigned by a learner to the pattern.

Figure 5 provides this information for several different rule patterns. Additionally, these plots include the empirical type (red dotted) and token proportions (red double-dashed) for the input corpus. Note how for the two major patterns, all models successfully track the type, rather than the token frequency, correctly developing a preference for stress-initial over unstressed words, despite the comparable token frequency of these two patterns. This is compatible with a recent proposal by Thiessen and Saffran (2007), who argue that infants infer the

stress pattern over their lexicon. For a Bayesian model such as ours or Goldwater et al. (2009)’s, there is no need to pre-specify that the distribution ought to be learned over types rather than tokens, as the models automatically interpolate between type and token statistics according to the properties of their input (Goldwater et al., 2006). In addition, a Bayesian framework provides a simple answer to the question of how a learner might identify the role of stress in its language without already having acquired at least some words. By combining different kinds of cues, e.g. distributional, phonotactic and prosodic, in a principled manner a Bayesian learner can jointly segment its input and learn the appropriate role of each cue, without having to pre-specify specific preferences that might differ across languages.

The iambic rule pattern that puts stress on the second syllable is much more infrequent on a token level. All models track this low token frequency, underestimating the type frequency of this pattern by a fair amount. This suggests that learning this pattern correctly requires considerably more input than for the other patterns. Indeed, the iambic pattern is known to pose problems for infants when they start using stress as an effective cue. It is only from roughly 10 months of age that infants successfully segment iambic words (Jusczyk et al., 1999b). Not surprisingly, the USC doesn’t aid in learning about this pattern because it is completely silent on where stress might fall (and does not noticeably improve segmentation performance to begin with).

Finally, we can also investigate whether the models that lack the USC nevertheless learn that words contain at most one lexically stressed syllable. The bottom-right graph in Figure 5 plots the probability assigned by the models to patterns that violate the USC. This includes, for example, the rules $\underline{\text{Word}} \rightarrow \text{SyllS SyllS}$ and $\underline{\text{Word}} \rightarrow \text{SyllS SyllU SyllS}$. Note how the probabilities assigned to these rules approaches zero, indicating that the learner becomes more certain that there are no words that contain more than one syllable with lexical stress. As we argued above, this suggests that a Bayesian learner can acquire the USC from a modest amount of data — it will properly infer that the unnatural patterns are simply not supported by the input. To summarize, by examining the internal

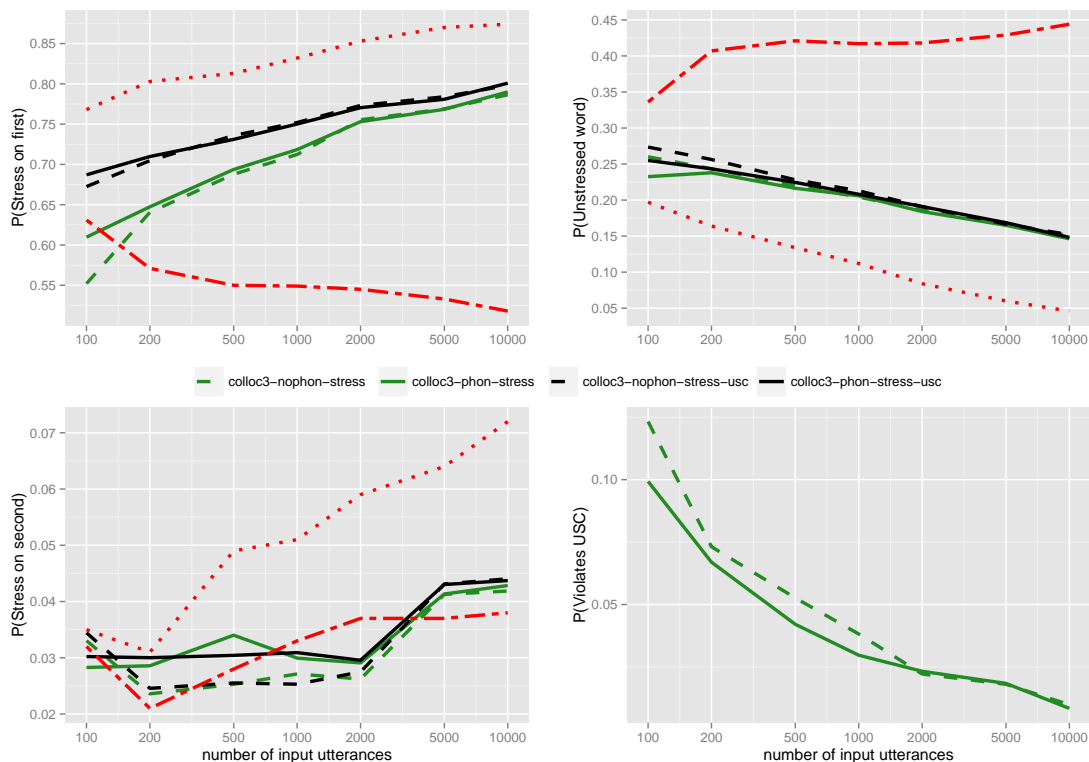


Figure 5: Evolution of the knowledge the learner acquires on the Alex corpus. The red dotted line indicates the empirical type distribution of a specific pattern, and the double-dashed line the empirical token distribution. Top-Left: Stress-initial pattern, Top-Right: Unstressed Words, Bottom-Left: Stress-second pattern, Bottom-Right: Patterns that violate the USC.

state of the Bayesian learners we can characterise how their knowledge about the stress preferences of their languages develops, rather than merely measuring how well they perform word segmentation. We find that the iambic pattern that has been observed to pose problems for infant learners also is harder for the Bayesian learner to acquire, arguably due to its extremely low token-frequency.

6 Conclusion and Future Work

We have presented Adaptor Grammar models of word segmentation that are able to take advantage of stress cues and are able to learn from phonemic input. We find that phonotactics and stress interact in interesting ways, and that stress cues makes a stable contribution to existing word segmentation models, improving their performance by 4-6% token f-score. We also find that the USC introduced by Yang (2004) need not be prebuilt into a model but can be acquired by a Bayesian learner from the data. Similarly, we directly investigate the stress preferences

acquired by our models and find that for stress-initial and unstressed words, they track type rather than token frequencies. The rare stress-second pattern seems to require more input to be properly acquired, which is compatible with infant development data.

An important goal for future research is to evaluate segmentation models on typologically different languages and to study the relative usefulness of different cues cross-lingually. For example, languages such as French lack lexical stress; it would be interesting to know whether in such a case, phonotactic (or other) cues are more important. Relatedly, recent work such as Börschinger et al. (2013) has found that artificially created data often masks the complexity exhibited by real speech. This suggests that future work should use data directly derived from the acoustic signal to account for contextual effects, rather than using dictionary look-up or other heuristics. In using the Alex corpus, for which good quality audio is available, we have taken a first step in this direction.

Acknowledgements

This research was supported by the Australian Research Council's Discovery Projects funding scheme (project numbers DP110102506 and DP110102593). We'd like to thank Professor Dupoux and our other colleagues at the Laboratoire de Sciences Cognitives et Psycholinguistique in Paris for hosting us while this research was performed, as well as the Mairie de Paris, the Fondation Pierre Gilles de Gennes, the Ecole des Hautes Etudes en Sciences Sociales, the Ecole Normale Supérieure, The Region Ile de France, the European Research Council (ERC-2011-AdG-295810 BOOTPHON), the Agence Nationale pour la Recherche (ANR-2010-BLAN-1901-1 BOOTLANG, ANR-10-IDEX-0001-02 and ANR-10-LABX-0087) and the Fondation de France. We'd also like to thank three anonymous reviewers for helpful comments and suggestions.

References

- N. Bernstein-Ratner. 1987. The phonology of parent-child speech. In K. Nelson and A. van Kleeck, editors, *Children's Language*, volume 6. Erlbaum, Hillsdale, NJ.
- Benjamin Börschinger, Katherine Demuth, and Mark Johnson. 2012. Studying the effect of input size for Bayesian word segmentation on the Providence corpus. In *Proceedings of the 24th International Conference on Computational Linguistics (Coling 2012)*, pages 325–340. Coling 2012 Organizing Committee.
- Benjamin Börschinger, Mark Johnson, and Katherine Demuth. 2013. A joint model of word segmentation and phonological variation for English word-final /t/-deletion. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1508–1516. Association for Computational Linguistics.
- M. Brent. 1999. An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 34:71–105.
- M. Christiansen and S. Curtin. 1999. The power of statistical learning: No need for algebraic rules. In *Proceedings of the 21st Annual Conference of the Cognitive Science Society*.
- Morten H Christiansen, Joseph Allen, and Mark S Seidenberg. 1998. Learning to segment speech using multiple cues: A connectionist model. *Language and Cognitive Processes*, 13(2-3):221–268.
- Suzanne Curtin, Toben H Mintz, and Morten H Christiansen. 2005. Stress changes the representational landscape: Evidence from word segmentation. *Cognition*, 96(3):233–262.
- Anne Cutler and David M Carter. 1987. The predominance of strong initial syllables in the English vocabulary. *Computer Speech and Language*, 2(3):133–142.
- Anne Cutler, Jacques Mehler, Dennis Norris, and Juan Segui. 1986. The syllable's differing role in the segmentation of French and English. *Journal of Memory and Language*, 25(4):385 – 400.
- Anne Cutler. 2005. Lexical stress. In David B. Pisoni and Robert E. Remez, editors, *The Handbook of Speech Perception*, pages 264–289. Blackwell Publishing.
- K. Demuth, J. Culbertson, and J. Alter. 2006. Word-minimality, epenthesis, and coda licensing in the acquisition of English. *Language and Speech*, 49:137–174.
- Gabriel Doyle and Roger Levy. 2013. Combining multiple information types in Bayesian word segmentation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 117–126. Association for Computational Linguistics.
- Victoria Fromkin, editor. 2001. *Linguistics: An Introduction to Linguistic Theory*. Blackwell, Oxford, UK.
- Sharon Goldwater, Tom Griffiths, and Mark Johnson. 2006. Interpolating between types and tokens by estimating power-law generators. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 459–466. MIT Press.
- Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2009. A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112(1):21–54.
- Sharon Goldwater. 2007. *Nonparametric Bayesian Models of Lexical Acquisition*. Ph.D. thesis, Brown University.
- Mark Johnson and Katherine Demuth. 2010. Unsupervised phonemic Chinese word segmentation using Adaptor Grammars. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 528–536. Coling 2010 Organizing Committee.
- Mark Johnson and Sharon Goldwater. 2009. Improving nonparametric Bayesian inference: experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational*

- Linguistics*, pages 317–325. Association for Computational Linguistics.
- Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007. Adaptor Grammars: A framework for specifying compositional nonparametric Bayesian models. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 641–648. MIT Press, Cambridge, MA.
- Mark Johnson. 2008a. Unsupervised word segmentation for Sesotho using Adaptor Grammars. In *Proceedings of the Tenth Meeting of ACL Special Interest Group on Computational Morphology and Phonology*, pages 20–27. Association for Computational Linguistics.
- Mark Johnson. 2008b. Using Adaptor Grammars to identify synergies in the unsupervised acquisition of linguistic structure. In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics*, pages 398–406. Association for Computational Linguistics.
- Peter W Jusczyk, Anne Cutler, and Nancy J Redanz. 1993. Infants’ preference for the predominant stress patterns of English words. *Child Development*, 64(3):675–687.
- Peter W. Jusczyk, E. A. Hohne, and A. Bauman. 1999a. Infants’ sensitivity to allophonic cues for word segmentation. *Perception and Psychophysics*, 61:1465–1476.
- Peter W. Jusczyk, Derek M. Houston, and Mary Newsome. 1999b. The beginnings of word segmentation in English-learning infants. *Cognitive Psychology*, 39(3-4):159–207.
- Peter Jusczyk. 1997. *The discovery of spoken language*. MIT Press, Cambridge, MA.
- Myron Korman. 1984. Adaptive aspects of maternal vocalizations in differing contexts at ten weeks. *First Language*, 5:44–45.
- Constantine Lignos and Charles Yang. 2010. Recession segmentation: simpler online word segmentation using limited resources. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 88–97. Association for Computational Linguistics.
- Constantine Lignos. 2011. Modeling infant word segmentation. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 29–38. Association for Computational Linguistics.
- Constantine Lignos. 2012. Infant word segmentation: An incremental, integrated model. In *Proceedings of the West Coast Conference on Formal Linguistics 30*.
- Brian MacWhinney. 2000. The CHILDES project: Tools for analyzing talk: Volume I: Transcription format and programs, volume II: The database. *Computational Linguistics*, 26(4):657–657.
- Sven L Mattys and Peter W Jusczyk. 2000. Phonotactic cues for segmentation of fluent speech by infants. *Cognition*, 78(2):91–121.
- Sven L Mattys. 2000. The perception of primary and secondary stress in English. *Perception and Psychophysics*, 62(2):253–265.
- Lisa Pearl, Sharon Goldwater, and Mark Steyvers. 2011. Online learning mechanisms for Bayesian models of word segmentation. *Research on Language and Computation*, 8(2):107–132.
- Elisabeth O. Selkirk. 1984. *Phonology and Syntax: The Relation Between Sound and Structure*. MIT Press.
- Erik D Thiessen and Jenny R Saffran. 2003. When cues collide: use of stress and statistical cues to word boundaries by 7-to 9-month-old infants. *Developmental Psychology*, 39(4):706.
- Erik D Thiessen and Jenny R Saffran. 2007. Learning to learn: Infants acquisition of stress-based strategies for word segmentation. *Language Learning and Development*, 3(1):73–100.
- Carnegie Mellon University. 2008. The CMU pronouncing dictionary, v.0.7a.
- Charles Yang. 2004. Universal grammar, statistics or both? *Trends in Cognitive Sciences*, 8(10):451–456.

FLORS: Fast and Simple Domain Adaptation for Part-of-Speech Tagging

Tobias Schnabel

Department of Computer Science
Cornell University
tbs49@cornell.edu

Hinrich Schütze

Center for Information & Language Processing
University of Munich
inquiries@cislmu.org

Abstract

We present FLORS, a new part-of-speech tagger for domain adaptation. FLORS uses robust representations that work especially well for unknown words and for known words with unseen tags. FLORS is simpler and faster than previous domain adaptation methods, yet it has significantly better accuracy than several baselines.

1 Introduction

In this paper we describe FLORS, a part-of-speech (POS) tagger that is Fast in training and tagging, uses LOcal context only (as opposed to finding the optimal tag sequence for the entire sentence), performs Robustly on target domains (TDs) in unsupervised domain adaptation (DA) and is Simple in architecture and feature representation.

FLORS constructs a robust representation of the local context of the word v that is to be tagged. This representation consists of distributional features, suffixes and word shapes of v and its local neighbors. We show that it has two advantages.

First, since the main predictors used by FLORS are distributional features (not the word’s identity), FLORS *predicts unseen tags of known words better* than prior work on DA for POS. Second, since FLORS uses representations computed from unlabeled text, representations of unknown words are in principle of the same type as representations of known words; this property of FLORS results in *better performance on unknown words* compared to prior work. These two advantages are especially beneficial for TDs that contain high rates of unseen tags of known words and high rates of unknown

words. We show that FLORS achieves excellent DA tagging results on the five domains of the SANCL 2012 shared task (Petrov and McDonald, 2012) and outperforms three state-of-the-art taggers on Blitzer et al.’s (2006) biomedical data.

FLORS is also simpler and faster than other POS DA methods. It is simple in that the input representation consists of three simple types of features: distributional count features and two types of binary features, suffix and shape features. Many other word representations that are used for improving generalization (e.g., (Brown et al., 1992; Collobert et al., 2011)) are costly to train or have difficulty handling unknown words. Our representations are fast to build and can be created on-the-fly for unknown words that occur during testing.

The learning architecture is simple and fast as well. We train k binary one-vs-all classifiers that use local context only and no sequence information (where k is the number of tags). Thus, tagging complexity is $O(k)$. Many other learning setups for DA are more complex; e.g., they *learn* representations (as opposed to just counting), they learn *several classifiers* for different subclasses of words (e.g., known vs. unknown) or they combine *left-to-right and right-to-left* taggings.

The next two sections describe experimental data, setup and results. Results are discussed in Section 4. We compare FLORS to alternative word representations in Section 5 and to related work in Section 6. Section 7 presents our conclusions.

2 Experimental data and setup

Data. Our source domain is the Penn Treebank (Marcus et al., 1993) of Wall Street Journal (WSJ)

text. Following Blitzer et al. (2006), we use sections 2-21 for training and 100,000 WSJ sentences from 1988 as unlabeled data in training.

We evaluate on six different TDs. The first five TDs (newsgroups, weblogs, reviews, answers, emails) are from the SANCL shared task (Petrov and McDonald, 2012). Additionally, the SANCL dataset contains sections 22 and 23 of the WSJ for in-domain development and testing, respectively. Each SANCL TD has an unlabeled training set of 100,000 sentences and development and test sets of about 1000 labeled sentences each. The sixth TD is BIO, the Penn BioTreebank data set distributed by Blitzer. It consists of dev and test sets of 500 sentences each and 100,000 unlabeled sentences.

Classification setup. Similar to SVMTool (Giménez and Màrquez, 2004) and Choi and Palmer (2012) (henceforth: C&P), we use *local context only* for tagging instead of performing sequence classification. For a word w occurring as token v_i in a sentence, we build a feature vector for a local window of size $2l + 1$ around v_i . The representation of the object to be classified is this feature vector and the target class is the POS tag of v_i .

We use the linear L2-regularized L2-loss SVM implementation provided by LIBLINEAR (Fan et al., 2008) to train k one-vs-all classifiers on the training set where k is the number of POS tags in the training set (in our case $k = 45$). We train with untuned default parameters; in particular, $C = 1$. In the special case of linear SVMs, the value of C does not need to be tuned exhaustively as the solution remains constant after C has reached a certain threshold value C^* (Keerthi and Lin, 2003). Training can easily be parallelized by giving each binary SVM its own thread.

Windows. The local context for tagging token v_i is a window of size $2l + 1$ centered around v_i : $(v_{i-l}, \dots, v_i, \dots, v_{i+l})$. We pad sentences on either side with $\langle \text{BOUNDARY} \rangle$ to ensure sufficient context for all words. Given a mapping f from words to feature vectors (see below), the representation F of a token v_i is the concatenation of the $2l + 1$ word vectors in its window

$$F(v_i) = f(v_{i-l}) \oplus \dots \oplus f(v_{i+l})$$

where \oplus is vector concatenation.

Word features. We represent each word w by four components: (i) counts of left neighbors, (ii) counts of right neighbors, (iii) binary suffix features and (iv) binary shape features. These four components are concatenated:

$$f(w) = f_{\text{left}}(w) \oplus f_{\text{right}}(w) \oplus f_{\text{suffix}}(w) \oplus f_{\text{shape}}(w)$$

We consider these sources of information equally important and normalize each of the four component vectors to unit length. Normalization also has a beneficial effect on SVM training time because it alleviates numerical problems (Fan et al., 2008).

Distributional features. We follow a long tradition of older (Finch and Chater, 1992; Schütze, 1993; Schütze, 1995) and newer (Huang and Yates, 2009) work on creating distributional features for POS tagging based on local left and right neighbors.

Specifically, the i^{th} entry x_i of $f_{\text{left}}(w)$ is the weighted number of times that the *indicator word* c_i occurs immediately to the left of w :

$$x_i = \text{tf}(\text{freq}(\text{bigram}(c_i, w)))$$

where c_i is the word with frequency rank i in the corpus, $\text{freq}(\text{bigram}(c_i, w))$ is the number of times the bigram “ $c_i w$ ” occurs in the corpus and we weight the non-zero frequencies logarithmically: $\text{tf}(x) = 1 + \log(x)$. tf-weighting has been used by other researchers (Huang and Yates, 2009) and showed good performance in our own previous work.

$f_{\text{right}}(w)$ is defined analogously. We restrict the set of indicator words to the $n = 500$ most frequent words in the corpus. To avoid zero vectors, we add an entry x_{n+1} to each vector that counts omitted contexts:

$$x_{n+1} = \text{tf} \left(\sum_{j:j>n} \text{freq}(\text{bigram}(c_j, w)) \right)$$

We compute distributional vectors on the joint corpus \mathcal{D}_{ALL} of all labeled and unlabeled text of source domain and TD. The text is preprocessed by lowercasing everything – which is often done when computing word representations, e.g., by Turian et al. (2010) – and by padding sentences with $\langle \text{BOUNDARY} \rangle$ tokens.

Suffix features. Suffixes are promising for DA because basic morphology rules are the same in different domains. In contrast to other work on tagging

	model	classifier	features
1	TnT	HMM	$p_{-\{0,1,2\}}$, v_0 , suffixes (for OOVs)
2	Stanford	bidir. MEMM	$p_{\pm\{0,1,2\}}$, $v_{\pm\{0,1\}}$, affixes, orthography
3	SVMTool	SVM	$p_{\pm\{0,1,2,3\}}$, $v_{\pm\{0,1,2,3\}}$, affixes, orthography, word length
4	C&P	SVM	$p_{\pm\{0,1,2,3\}}$, $v_{\pm\{0,1,2,3\}}$, affixes, orthography
5	FLORS	SVM	distributions of $v_{\pm\{0,1,2\}}$, suffixes, orthography

Table 1: Overview of baseline taggers and FLORS. v_i : token, p_i : POS tag. Positions included in the sets of token indices are relative to the position i of the word v_0 to be tagged; e.g., $p_{\pm\{0,1,2\}}$ is short for $\{p_{-0}, p_{-1}, p_{-2}, p_0, p_1, p_2\}$. To represent tokens v_i , models 1–4 use vocabulary indices and FLORS uses distributional representations. Models 2–4 use combinations of features (e.g., tag-word) as well.

(e.g., Ratnaparkhi (1996), Toutanova et al. (2003), Miller et al. (2007)) we simply use *all (lowercase) suffixes* to avoid the need for selecting a subset of suffixes; and we treat *all words* equally as opposed to using suffix features for only a subset of words. For suffix s , we set the dimension corresponding to s in $f_{\text{suffix}}(w)$ to 1 if lowercased w ends in s and to 0 otherwise. Note that w is a suffix of itself.¹

Shape features. We use the Berkeley parser word signatures (Petrov and Klein, 2007). Each word is mapped to a bit string encompassing 16 binary indicators that correspond to different orthographic (e.g., does the word contain a digit, hyphen, uppercase character) and morphological (e.g., does the word end in -ed or -ing) features. There are 50 unique signatures in WSJ. We set the dimension of $f_{\text{shape}}(w)$ that corresponds to the signature of w to 1 and all other dimensions to 0. We note that the shape features we use were designed for English and probably would have to be adjusted for other languages.

Baselines. We address the problem of unsupervised domain adaptation for POS tagging. For this problem, we consider three types of baselines: (i) high-performing publicly available systems, (ii) the taggers used at SANCL and (iii) POS DA results published for BIO.

Most of our experiments use taggers from category (i) because we can ensure that experimental conditions are directly comparable. The four baselines in category (i) are shown in Table 1. Three have near state-of-the-art performance on WSJ: SVMTool (Giménez and Màrquez, 2004), Stanford

(Toutanova et al., 2003) (a bidirectional MEMM) and C&P. TnT (Brants, 2000) is included as a representative of fast and simple HMM taggers. In addition, C&P is a tagger that has been extensively tested in DA scenarios with excellent results. Unless otherwise stated, we train all models using their default configuration files. We use the optimized parameter configuration published by C&P for the C&P model.

Test set results will be compared with the SANCL taggers (category (ii)) at the end of Section 3.

As far as category (iii) is concerned, most work on POS DA has been evaluated on BIO. We discuss our concerns about the BIO evaluation sets in Section 4, but also show that FLORS beats previously published results on BIO as well (see Table 6).

3 Experimental results

We train k binary SVM classifiers on the training set. A token in the test set is classified by building its feature vector, running the classifiers on it and then assigning it to the POS class whose one-vs-all LIBLINEAR classifier returns the largest score.

Results for ALL accuracy (accuracy for all tokens) and OOV accuracy (accuracy for tokens not occurring in the labeled WSJ data) are reported in Table 2. Results with an asterisk are significantly worse than a column’s best result using McNemar’s test ($p < .001$). We use the same test and p-value throughout this paper.

The basic FLORS model (Table 2, line 5) uses window size 5 ($l = 2$). Each word in the window has 1002 distributional features (501 left and right), 91,161 suffix features and 50 shape features. The final feature vector for a token has a dimensionality of about 500,000, but is very sparse.

FLORS outperforms all baselines on the five TDs

¹One could also compute these suffixes for $_w$ (w prefixed by underscore) instead of for w to include words as distinguishable special suffixes. We test this alternative in Table 2, line 15.

		newsgroups		reviews		weblogs		answers		emails		wsj		
		ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV	
1	TnT	88.66*	54.73*	90.40*	56.75*	93.33*	74.17*	88.55*	48.32*	88.14*	58.09*	95.75*	88.30	
2	Stanford	89.11*	56.02*	91.43*	58.66*	94.15*	77.13*	88.92*	49.30*	88.68*	58.42*	96.83	90.25	
3	SVMTool	89.14*	53.82*	91.30*	54.20*	94.21*	76.44*	88.96*	47.25*	88.64*	56.37*	96.63	87.96	
4	C&P	89.51*	57.23*	91.58*	59.67*	94.41*	78.46*	89.08*	48.46*	88.74*	58.62*	96.78	88.65	
5	FLORS	basic	90.86	66.42	92.95	75.29	94.71	83.64	90.30	62.15	89.44*	62.61	96.59	90.37
6		$n = 250$	90.93	67.03	92.93	75.45	94.69	83.69	90.29	62.20	89.63	63.43	96.56*	89.45
7		$n = 0$	89.14*	55.59*	91.80*	66.31*	93.40*	72.55*	89.47*	55.82*	88.21*	57.83*	96.29*	85.55*
8		no suffixes	90.60	65.17	92.74	71.94*	94.77	84.92	89.77*	58.71*	89.30*	62.09	96.28*	88.88
9		no shapes	89.70*	63.10*	92.24*	68.70*	92.60*	74.72*	89.55*	59.08*	89.63	64.17	95.52*	83.94*
10		$n = 0$	90.61*	65.95	92.76*	75.56	94.62	84.62	90.23	61.87	89.40*	63.82	96.51*	90.02
11		no suffixes	90.66	64.78*	92.88	75.08	94.83	84.52	90.36	61.92	89.42	62.74	96.64	89.45
12		no shapes	90.74	67.03	93.02	75.88	94.57	83.83	90.23	61.73	89.41*	63.49	96.57	90.25
13		$l = 1$	90.44*	63.62*	92.69*	75.72	94.48*	84.03	90.02*	62.66	89.17*	62.71	96.44*	88.65
14		L-to-R	90.56*	66.08	92.97	75.40	94.57	83.79	90.43	62.80	89.43	63.13	96.53*	90.94
15		voc. indices	90.93	66.64	92.91	75.03	94.71	84.08	90.27	61.92	89.37*	62.26	96.63	90.60

Table 2: Tagging accuracy of four baselines and FLORS on the dev sets. The table is structured as follows: baselines (lines 1–4), basic FLORS setup (lines 5–6), effect of omitting one of the three feature types if the word to be tagged is changed compared to the basic FLORS setup (lines 7–9) and if the word to be tagged is not changed compared to basic FLORS (lines 10–12), effect of three important configuration choices on tagging accuracy: window size (line 13), inclusion of prior tagging decision (line 14) and vocabulary index (line 15). n : number of indicator words. $2l + 1$: size of the local context window. Lines 10–12: Only the neighbors of v_0 are modified compared to basic (line 5). Lines 7–9: All five token representations (including v_0) are modified. A column’s best result is bold.

(line 5 vs. lines 1–4). Only in-domain on WSJ, three baselines are slightly superior. The baselines are slightly better on ALL accuracy because they were designed for tagging in-domain data and use feature sets that have been found to work well on the source domain. Generally, C&P performs best for DA among the baselines. On answers and WSJ, however, Stanford has better overall accuracies. These results are in line with C&P.

On lines 6–15, we investigate how different modifications of the basic FLORS model affect performance. First, we examine the effect of leaving out components of the representation: distributional features ($f_{\text{left}}(w)$, $f_{\text{right}}(w)$), suffixes ($f_{\text{suffix}}(w)$) and shape features ($f_{\text{shape}}(w)$).

Distributional features boost performance in all domains: ALL and OOV accuracies are consistently worse for $n = 0$ (line 7) than for $n \in \{250, 500\}$ (lines 6&5). FLORS with $n = 250$ has better OOV accuracies in 5 of 6 domains. However, ALL accuracy for FLORS with $n = 500$ is better in the majority of domains. The main result of this comparison is that FLORS does not seem to be very sensitive to the value of n if n is large enough.

Shape features also improve results in all do-

main, with one exception: emails (lines 9 vs 5). For emails, shape features decrease ALL accuracy by .19 and OOV accuracy by 1.56. This may be due to the fact that many OOVs are NNP/NN and that tagging conventions for NNP/NN vary between domains. See Section 4 for discussion.

Performance benefits from suffixes in all domains but weblogs (lines 8 vs 5). Weblogs contain many foreign names such as *Abdul* and *Yasim*. For these words, shapes apparently provide better information for classification than suffixes. ALL accuracies suffer little when leaving out suffixes, but the feature space is much smaller: about 3000 dimensions. Thus, for domains where we expect few OOVs, omitting suffix features could be considered.

Lines 7–9 omit one of the components of $f(v_i)$ for all five words in the local context: $i \in \{-2, -1, 0, 1, 2\}$. Lines 10–12 omit the same components for the neighbor words only – i.e., $i \in \{-2, -1, 1, 2\}$ – and leave $f(v_0)$ unchanged. 14 of the 6×3 ALL accuracies on lines 10–12 are worse than FLORS basic, 4 are better. The largest differences are .25 for newsgroups and .19 for reviews (lines 5 vs 10), but differences for the other domains are negligible. This shows that the most important

feature representation is that of v_0 (not surprisingly) and that the distributional features of the other words can be omitted at the cost of some loss in accuracy if a small average number of active features is desired.

Another FLORS parameter is the size of the local context. Surprisingly, OOV accuracies benefit a bit in four domains if we reduce l from 2 to 1 (lines 13 vs 5). However, ALL accuracy consistently drops in all six domains. This argues for using $l = 2$, i.e., a window size of 5.

Results for left-to-right (L-to-R) tagging are given on line 14. Similar to SVMTool and C&P, each sentence is tagged from left to right and previous tagging decisions are used for the current classification. In this setting, we use the previous tag p_{i-1} as one additional feature in the feature vector of v_i .

The effect of left-to-right is similar to the effect of omitting suffixes: OOV accuracies go up in some domains, but ALL accuracies decrease (except for an increase of .02 for reviews). This is in line with the experiments in (Schnabel and Schütze, 2013) where sequential information in a CRF was not robust across domains. OOV tagging may benefit from correct previous tags because the larger left context that is indirectly made available by left-to-right tagging compensates partially for the lack of information about the OOV word.

In contrast to standard approaches to POS tagging, the FLORS basic representation does not contain vocabulary indices. Line 15 shows what happens if we add them; the dimensionality of the feature vector is increased by $5|V|$ – where V is the training set vocabulary – and in training one binary feature is set to one for each of the five local context words. Performance is almost indistinguishable from FLORS basic, suggesting that only using suffixes – which can be viewed as “ambiguous” vocabulary indices, e.g., “at” is on for “at”, “mat”, “hat”, “laundromat” etc – is sufficient.

In summary, we find that distributional features, word signatures and suffixes all contribute to successful POS DA. Factors with only minor impact on performance are the number of indicator words used for the distributional representations, the window size l and the tagging scheme (L-to-R vs. non-L-to-R). Unknown words and known words behave differently with respect to certain feature choices.

The different behavior of unknown and known

words suggests that training and optimizing two separate models – an approach used by SVMTool – would further increase tagging accuracy. Note that there has been at least one publication (Schnabel and Schütze, 2013) on optimizing a separate model for unknown words that has in some cases better performance on OOV accuracy than what we publish here.² However, this would complicate the architecture of FLORS. We opted for a maximally simple model in this paper, potentially at the cost of some performance.

Test set results. Table 3 reports results on the test sets. FLORS again performs significantly better on all five TDs, both on ALL and OOV. Only in-domain on WSJ, ALL performance is worse.

Finally, we compare our results to the POS taggers for which performance was reported at SANCL 2012 (Petrov and McDonald, 2012, Table 4). Constituency-based parsers – which also tag words as a by-product of deriving complete parse trees – are excluded from the comparison because they are trained on a richer representation, the syntactic structure of sentences.³ FLORS’ results are better than the best non-parsing-based results at SANCL 2012, which were accuracies of 92.32 on newsgroups (HIT), 90.65 on reviews (HIT) and 91.07 on answers (IMS-1).

4 Discussion

Advantages of FLORS representation. As we can see in Table 1, the main representational difference between FLORS and the other taggers is that the FLORS representation does not include vocabulary indices of the word to be tagged or its neighbors – the FLORS vector only consists of distributional, suffix and shape features.

This is an obvious advantage for OOVs. In other representational schemes, OOVs have representations that are fundamentally different from known

²Schnabel and Schütze (2013) report OOV accuracies of 56.62 (newsgroups), 64.61 (reviews), 71.86 (weblogs), 54.28 (answers), 61.05 (emails) and 64.64 (BIO) for their basic model and even higher OOV accuracies if parameters are optimized on a per-domain basis.

³DCU-Paris13 is listed in the dependency parser tables, but DCU-Paris13 results are derived from a constituency parser. DCU also developed sophisticated preprocessing rules for the different domains, which can be viewed as a kind of manual domain adaptation.

		newsgroups		reviews		weblogs		answers		emails		wsj	
		ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV
1	TnT	90.85*	56.60*	89.67*	50.98*	91.37*	62.65*	89.36*	51.82*	87.38*	55.12*	96.57*	86.27
2	Stanford	91.25*	57.96*	90.30*	51.87*	92.32*	67.85*	89.74*	53.41*	87.77*	57.10*	97.43	88.71
3	SVMTool	91.21*	54.40*	90.01*	45.05*	92.05*	63.59*	89.90*	51.07*	87.74*	53.23*	97.26	86.47
4	C&P	91.68*	60.58*	90.42*	51.12*	92.22*	66.91*	89.90*	53.31*	87.91*	54.47*	97.44	88.20
5	FLORS basic	92.41	66.91	92.25	70.87	93.14	75.32	91.17	67.93	88.67	61.09	97.11*	87.79

Table 3: Tagging accuracy of four baselines and FLORS on the test sets.

		newsgroups	reviews	weblogs	answers	emails	wsj	bio
pct tokens	unknown tag	0.31	0.06	0.00	0.25	0.80	0.00	0.98
	OOV	10.34	6.84	8.45	8.53	10.56	2.72	19.86
	unseen word+tag	2.44	2.22	1.46	2.91	3.47	0.61	2.50
accuracy on unseen word+tag	TnT	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Stanford	3.66	5.74	9.40	5.46	2.77	15.23	4.64
	SVMTool	0.00	0.16	0.00	0.00	0.10	0.00	0.00
	C&P	14.47	14.75	20.51	13.37	10.29	38.07	8.98
	FLORS basic	21.06	21.97	21.65	17.19	15.13	41.12	12.69

Table 4: Top: Percentage of unknown tags, OOVs and unseen word+tag combinations (i.e., known words tagged with unseen tags) in the dev sets. Bottom: Tagging accuracy on unseen word+tag.

words – since their vocabulary index does not occur in the training set and cannot be used for prediction. In contrast, given enough unlabeled TD data, FLORS represents known and unknown words in essentially the same way and prediction of the correct tag is easier. This explanation is supported by the experiments in Table 2: FLORS beats all other systems on OOVs – even in-domain on WSJ.

In our analysis we found that apart from better handling of OOVs there is a second beneficial effect of distributional representations: they facilitate the correct tagging of *known words occurring with tags unseen in the training set*, which we call *unseen word+tags*. Table 4 gives statistics on this case and shows that unseen word+tags occur at least two times as often out-of-domain (e.g., 1.46% for weblogs) than in-domain (.61% for WSJ). The bottom part of the table shows performance of the five taggers on unseen word+tags. FLORS is the top performer on all seven domains, with large differences of more than 5% in some domains.

The explanation is similar to the OOV case: FLORS does not restrict the set of possible POS’s of a word. The other taggers in Table 2 use the vocabulary index of the word to be tagged and will therefore give a strong preference to seen tags. Since FLORS

uses distributional features, it can more easily assign an unseen tag as long as it is compatible with the overall pattern of distribution, suffixes and shapes typical of the tag. C&P also perform relatively well on unseen word+tag due to the ambiguity classes in their model, but FLORS representations are better for every domain. We take these results to mean that constraints on a word’s possible POS tags may well be helpful for in-domain data, but for out-of-domain data an overly strong bias for a word’s observed tags is harmful.

It is important to stress that representations similar to FLORS representations have been used for a long time; we would expect many of them to have similar advantages for unseen word+tags. E.g., Brown clusters (Brown et al., 1992) and word embeddings (Collobert et al., 2011) are similar to FLORS in this respect. However, FLORS representations are extracted by simple counting whereas the computation of Brown clusters or word embeddings is much more expensive. The speed with which FLORS representations can be computed is particularly beneficial when taggers need to be adapted to new domains. FLORS can easily adapt its representations on the fly – as each new occurrence of a word is encountered, the counts that are the basis

for the x_i can simply be incremented. We present a direct comparison of FLORS representations with other representations in Section 5.

“Local context” vs. sequence classification. The most common approach to POS tagging is to tag a sentence with its most likely sequence; in contrast, independent tagging of local context is not guaranteed to find the best sequence. Recent work on English suggests that window-based tagging can perform as well as sequence-based methods (Liang et al., 2008; Collobert et al., 2011). Toutanova et al. (2003) report similar results. In our experiments, we also did not find consistent improvements when we incorporated sequence constraints (Table 2, line 14). However, there may be languages and applications involving long-distance relationships where local-context classification is suboptimal.

Local-context classification has two advantages compared to sequence classification. (i) It simplifies the classification and tagging setup: we can use any existing statistical classifier. Sequence classification limits the range of methods that can be applied; e.g., it is difficult to find a good CRF implementation that can handle real-valued features – which are of critical importance for our representation.

(ii) The time complexity of FLORS in tagging is $O(skf)$ where s is the length of the sentence, k is the number of tags and f is the number of non-zero features per local-context representation. In contrast, sequence decoding complexity is $O(sk^2f)$. This difference is not of practical importance for standard English POS sets, but it could be an argument against sequence classification for tagging problems with much larger tag sets.

In summary, replacing sequence classification with local-context classification is attractive for large-scale, practical tagging.

What DA can and cannot do. Despite the superior DA tagging results we report for FLORS in this paper, there is still a gap of 2%–7% (depending on the domain) between in-domain WSJ accuracy and DA accuracy on SANCL. In our analysis of this gap, we found some evidence that DA performance can be further improved – especially as more unlabeled TD data becomes available. But we also found two reasons for low performance that unsupervised DA cannot do anything about: differences in tag sets – or unknown tags – and differences in annotation guide-

lines.

Table 4 shows that *unknown tags* occur in five of the seven TDs at rates between 0% (weblogs) and 1% (BIO). Each token that is tagged with an unknown tag is necessarily an error in unsupervised DA. Furthermore, the unknown tag can also impact tagging accuracy in the local context⁴ – so the unknown tag rates in Table 4 are probably lower bounds for the error that is due to unknown tags. Based on these considerations, it is not surprising that tagging accuracy (e.g., of FLORS basic) and unknown tag rate are correlated as we can see in Tables 2, 4 and 6; e.g., we get the highest accuracies in the two domains that do not have unknown tags (weblogs and WSJ) and the lowest accuracy in the domain with the highest rate (BIO).

Since unknown tags cannot be predicted correctly, one could simply report accuracy on known tags. However, given the negative effect of unknown tags on tagging accuracy of the local context in which they occur, excluding unknown tags does not fully address the problem. For this reason, it is probably best to keep the common practice of simply reporting accuracy on all tokens, including unknown tags. But the percentages of unknown tags should also be reported for each dataset as a basis for a more accurate interpretation of results.

Another type of error that cannot be avoided in unsupervised DA is due to differences in *annotation guidelines*. There are a few such problems in SANCL; e.g., file names like “Services.doc” are annotated as NN in the email domain. But their distributional and grammatical behavior is more similar to NNPs; as a consequence, most file names are incorrectly tagged. In general, it is difficult to discriminate NNs from NNPs. The Penn Treebank annotation guidelines (Santorini, 1990) are compatible with either tag in many cases and it may simply be impossible to write annotation guidelines that avoid these problems (cf. Manning (2011)). NN-NNP inconsistencies are especially problematic for OOV tagging since most OOVs are NNs or NNPs.

⁴For example, there is a special tag ADD in the web domain for web addresses. The last two words of the sentence “I would like to host my upcoming website to/IN Liquidweb.com/ADD” are mistagged by Stanford tagger as “... to/TO Liquidweb.com/VB”. So the missing tag in this case also affects the tagging of surrounding words.

	bio dev		wsj train
	OOV	ALL	ALL
NN	62.4	25.4	14.4
JJ	15.9	8.9	6.2
NNS	10.2	7.5	6.3
NNP	0.5	0.2	9.5
NNPS	0.0	0.0	0.3

Table 5: Frequency of some tags (percent of tokens) for bio dev and wsj train.

While the amount of inconsistent annotation is limited for SANCL, it is a serious problem for BIO. Table 5 shows that the proportion of NNPs in BIO is less than a tenth of that in WSJ (.2 in BIO vs. 9.5 in WSJ). This is due to the fact that many bio-specific names, in particular genes, are annotated as NN. In contrast, the distributionally and orthographically most similar names in WSJ are tagged as NNP. For example, we find “One cell was teased out, and its DNA/NNP extracted” in WSJ vs. “DNA/NN was isolated” in BIO.

	standard setup		NNP→NN		
	ALL	OOV	ALL	OOV	
TnT	87.49*	59.08*	91.75*	78.33*	
Stanford	88.46*	62.55*	92.36*	79.19*	
SVMTool	88.33*	61.30*	92.47	79.46*	
C&P	87.82*	60.60*	92.06*	79.30*	
FLORS	basic	88.90	64.74	92.91	82.58
	$n = 250$	88.90	64.51	92.93	82.47
	$n = 0$	87.27*	57.75*	90.91*	73.57*
	no suffixes	88.09*	62.20*	91.98*	79.27*
	no shapes	87.78*	59.82*	91.81*	77.31*
	$l = 1$	89.12	65.52	92.99	82.90

Table 6: Tagging accuracy on bio dev. NNP→NN results were obtained by replacing NNPs with NNs.

Given this large discrepancy in the frequency of the tag NNP – which arguably is due to different annotation guidelines, not due to underlying differences between the two genres – BIO should probably not be used for evaluating DA. This is why we did not include it in our comparison in Table 2.

For sake of completeness, we provide tagging accuracies for BIO in Table 6, “standard setup”. The results are in line with SANCL results: FLORS beats the baselines on ALL and OOV accuracies.

However, if we build the NN bias into our model by simply replacing all NNP tags with NN tags, then accuracy goes up by 4% on ALL and by almost 20% on OOV. Even TnT, the most basic tagger, achieves ALL/OOV accuracy of 91.75/78.33, better than any method in the standard setup. These accuracies are well above those in (Blitzer et al., 2006) and (Huang and Yates, 2010).

Since simply replacing NNPs with NNs has such a large effect, BIO cannot be used sensibly for evaluating DA methods. In practice, it is not possible to separate “true” improvements due to generic better DA from elements of the proposed method that simply introduce a negative bias for NNP.

In summary, when comparing different DA methods caution should be exercised in the choice of domains. In particular, the effect of unknown tags should be made transparent and the gold standards should be analyzed to determine whether the task addressed in the TD differs significantly in some aspects from that addressed in the source domain.

5 Comparison of word representations

Our approach to DA is an instance of representation learning: we aim to find representations that are robust across domains. In this section, we compare FLORS with two other widely used representation learning methods: (i) Brown clusters (Brown et al., 1992) and (ii) C&W embeddings, the word embeddings of Collobert et al. (2011). We use $f_{\text{dist}}(w) = f_{\text{left}}(w) \oplus f_{\text{right}}(w)$ to refer to our own distributional word representations (see Section 2).

The perhaps oldest and most frequently used low-dimensional representation of words is based on Brown clusters. Typically, prefixes of Brown clusters (Brown et al., 1992) are added to increase the robustness of POS taggers (e.g., Toutanova et al. (2003)). Computational costs are high (quadratic in the vocabulary size) although the computation can be parallelized (Uszkoreit and Brants, 2008).

More recently, general word representations (Collobert et al., 2011; Turian et al., 2010) have been used for robust POS tagging. These word representations are typically trained on a large amount of unlabeled text and fine-tuned for specific NLP tasks. Similar to Brown clusters, they are low-dimensional and can be used as features in many NLP tasks, ei-

ther alone or in combination with other features.

To compare $f_{\text{dist}}(w)$ (our distributional representations) with Brown clusters, we induced 1000 Brown clusters on the joint corpus data \mathcal{D}_{ALL} (see Section 2) using the publicly available implementation of Liang (2005). We padded sentences with $\langle \text{BOUNDARY} \rangle$ tokens on each side and used path prefixes of length 4, 6, 10 and 20 as features for each word (cf. Ratinov and Roth (2009), Turian et al. (2010)).

C&W embeddings are provided by Collobert et al. (2011): 50-dimensional vectors for 130,000 words from WSJ, trained on Wikipedia. Similar to our distributional representations $f_{\text{dist}}(w)$, the embeddings also contain a $\langle \text{BOUNDARY} \rangle$ token (which they call PADDING). Moreover, they have a special embedding for unknown words (called UNKNOWN) which we use whenever we encounter a word that is not in their lookup table. We preprocess our raw tokens the same way they do (lowercase and replace sequences of digits by “0”) before we look up a representation during training and testing.

We replaced the distributional features in our basic setup by either Brown cluster features or C&W embeddings. Table 7 repeats lines 5 and 7 of Table 2 and gives results of the modified FLORS setup.

All three representations improve both ALL and OOV accuracies in all domains. f_{dist} outperforms Brown in all cases except for OOV on emails. Brown may suffer from noisy data; cleaning methods have been used in the literature (Liang, 2005; Turian et al., 2010), but they are not unproblematic since a large part of the data available is lost, which results in more unknown words.

Brown and f_{dist} can be directly compared since they were trained on exactly the same data. f_{dist} and C&W are harder to compare directly because there are many differences. (i) C&W is trained on a much larger dataset. One consequence of this is that OOV accuracy on WSJ may be higher because some words that are unknown for other methods are actually known to C&W. (ii) C&W vectors are not trained on the SANCL TD data sets – this gives f_{dist} an advantage. (iii) C&W vectors are not trained on the WSJ. Again, this could give f_{dist} an advantage. (iv) C&W and f_{dist} are fundamentally different in the way they handle unknown words. C&W has a limited vocabulary and must replace all words not in

this vocabulary by the token UNKNOWN. In contrast, f_{dist} can create a meaningful individual representation for any OOV word it encounters.

Our FLORS tagger provides best ALL accuracies in all domains but WSJ, where C&W has best results. The good performance of C&W is rather unsurprising since the embeddings were created for the 130,000 most frequent words of the WSJ and thus cover the WSJ domain much better. Also, WSJ was used to tune parameters during development. As with our previous experiments, OOV results on emails seem slightly more sensitive to parameter choices than on other domains (recall the discussion of this issue in Section 4).

In summary, we have shown that f_{dist} representations work better for POS DA than Brown clusters. Furthermore, the evidence we have presented suggests that f_{dist} are comparable in performance to C&W embeddings if not better for POS DA.

The most important difference between f_{dist} and Brown / C&W is that f_{dist} are much simpler and much faster to compute. They are simpler because they are just slightly transformed counts in contrast to the other two approaches, which solve complex optimization problems. f_{dist} can be computed efficiently through simple incrementation in one pass through the corpus. In contrast, the other two approaches are an order of magnitude slower.

6 Related work

Unsupervised DA methods can be broadly put into four categories: representation learning and constraint-based frameworks – which require some tailoring to a task – and instance weighting and bootstrapping – which can be more generally applied to a wide range of problems. Since many approaches are application-specific, we focus on the ones that have been applied to POS tagging.

Representation learning. We already discussed two important approaches to representation learning in Section 5: C&W embeddings and Brown clusters.

Blitzer et al.’s (2006) structural correspondence learning (SCL) supports DA by creating similar representations for correlated features in the *pivot feature space*. This is a potentially powerful method. FLORS is simpler in that correlations are made directly accessible to the supervised learner.

	newsgroups		reviews		weblogs		answers		emails		wsj	
	ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV
1	90.86	66.42	92.95	75.29	94.71	83.64	90.30	62.15	89.44	62.61	96.59	90.37
2	89.14*	55.59*	91.80*	66.31*	93.40*	72.55*	89.47*	55.82*	88.21*	57.83*	96.29*	85.55*
3	90.57	64.57	92.54*	72.48*	94.51	80.58*	90.23	60.99	89.44	63.13	96.72	90.48
4	90.34*	62.41*	92.23*	71.47*	94.45	81.76	89.71*	56.28*	89.02*	63.20	96.48*	87.50

Table 7: Tagging accuracy of different word representations on the dev sets. Line 1 corresponds to FLORS basic. n : number of indicator words. A column’s best result is bold.

Moreover, FLORS representations consist of simple counts whereas SCL solves a separate optimization problem for each pivot feature.

Umansky-Pesin et al. (2010) derive distributional information for OOVs by running web queries. This approach is slow since it depends on a search engine.

Ganchev et al. (2012) successfully use search logs. This is a promising enhancement for FLORS.

Huang and Yates (2009) evaluate CRFs with distributional features. They examine lower dimensional feature representations using SVD or the latent states of an unsupervised HMM. They find better accuracies for their HMM method than Blitzer et al. (2006); however, they do not compare them against a CRF baseline using distributional features.

In later work, Huang and Yates (2010) add the latent states of multiple, differently trained HMMs as features to their CRF. Huang and Yates (2012) argue that finding an optimal feature representation is computationally intractable and propose a new framework that allows prior knowledge to be integrated into representation learning.

Latent sequence states are a form of word representation. Thus, it would be interesting to compare them to the non-sequence-based distributional representation that FLORS uses.

Constraint-based methods. Rush et al. (2012) use global constraints on OOVs to improve out-of-domain tagging. Although constraints ensure consistency, they require careful manual engineering. Distributional features can also be seen as a form of constraint since feature weights will be shared among all words.

Subramanya et al. (2010) construct a graph to encourage similar n-grams to be tagged similarly, resulting in moderate gains in one domain, but no gains on BIO when compared to self-training. The reason could be an insufficient amount of unsupervised data for BIO (100,000 sentences). Our ap-

proach does not seem to suffer from this problem.

Bootstrapping. Both self-training (McClosky et al., 2006) – which uses one classification model – and co-training (Blum and Mitchell, 1998) – which uses ≥ 2 models – have been applied to POS tagging.

Self-training usually improves a POS baseline only slightly if at all (Huang et al., 2009; Huang and Yates, 2010). Devising features based on labeled instances (instead of training on them) has been more successful (Florian et al., 2004; Sjøgaard, 2011).

Chen et al. (2011) use co-training for DA. In each round of their algorithm, both new training instances from the unlabeled data and new features are added. Their model is limited to binary classification. The co-training method of Kübler and Baucom (2011) trains several taggers and adds sentences from the TD to the training set on which they agree. They report slight, but statistically significant increases in accuracy for POS tagging of dialogue data.

Instance weighting. Instance weighting formalizes DA as the problem of having data from different probability distributions in each domain. The goal is to make these two distributions align by using instance-specific weights during training. Jiang and Zhai (2007) propose a framework that integrates prior knowledge from different data sets into the learning objective by weights.

In related work, C&P train *generalized* and *domain-specific* models. An input sentence is tagged by the model that is most similar to the sentence. FLORS could be easily extended along these lines, an experiment we plan for the future.

In terms of the basic classification setup, our POS tagger is most similar to the SVM-based approaches of Giménez and Màrquez (2004) and C&P. However, we do not use a left-to-right approach when tagging sentences. Moreover, SVMTool trains two separate models, one for OOVs and one for known words. FLORS only has a single model. In addition,

we do not make use of ambiguity classes, token-tag dictionaries and rare feature thresholds. Instead, we rely only on three types of features: distributional representations, suffixes and word shapes.

The local-context-only approach of SVMTool, C&P and FLORS is different from standard sequence classification such as MEMMs (e.g., Ratnaparkhi (1996), Toutanova et al. (2003), Tsuruoka and Tsujii (2005)) and CRFs (e.g., Collins (2002)). Sequence models are more powerful in theory, but this may not be an advantage in DA because the subtle dependencies they exploit may not hold across domains.

7 Conclusion

We have presented FLORS, a new POS tagger for DA. FLORS uses robust representations that work especially well for unknown words and for known words with unseen tags. FLORS is simpler and faster than previous DA methods, yet we were able to demonstrate that it has significantly better accuracy than several baselines.

Acknowledgments. This work was supported by DFG (Deutsche Forschungsgemeinschaft).

References

- John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*, pages 120–128.
- Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT*, pages 92–100.
- Thorsten Brants. 2000. TnT: A statistical part-of-speech tagger. In *ANLP*, pages 224–231.
- Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479.
- Minmin Chen, Kilian Q. Weinberger, and John Blitzer. 2011. Co-training for domain adaptation. In *NIPS*, pages 1–9.
- Jinho D. Choi and Martha Palmer. 2012. Fast and robust part-of-speech tagging using dynamic model selection. In *ACL: Short Papers*, pages 363–367.
- Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *EMNLP*, pages 1–8.
- Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.
- Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.
- Steven Finch and Nick Chater. 1992. Bootstrapping syntactic categories using statistical methods. In *Background and Experiments in Machine Learning of Natural Language*, pages 229–235.
- Radu Florian, Hany Hassan, Abraham Ittycheriah, Hongyan Jing, Nanda Kambhatla, Xiaoqiang Luo, Nicolas Nicolov, and Salim Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *HLT-NAACL*, pages 1–8.
- Kuzman Ganchev, Keith Hall, Ryan McDonald, and Slav Petrov. 2012. Using search-logs to improve query tagging. In *ACL: Short Papers*, pages 238–242.
- Jesús Giménez and Lluís Màrquez. 2004. SVMTool: A general pos tagger generator based on support vector machines. In *LREC*, pages 43–46.
- Fei Huang and Alexander Yates. 2009. Distributional representations for handling sparsity in supervised sequence-labeling. In *ACL-IJCNLP*, pages 495–503.
- Fei Huang and Alexander Yates. 2010. Exploring representation-learning approaches to domain adaptation. In *DANLP*, pages 23–30.
- Fei Huang and Alexander Yates. 2012. Biased representation learning for domain adaptation. In *EMNLP-CoNLL*, pages 1313–1323.
- Zhongqiang Huang, Vladimir Eidelman, and Mary Harper. 2009. Improving a simple bigram HMM part-of-speech tagger by latent annotation and self-training. In *NAACL-HLT: Short Papers*, pages 213–216.
- Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in NLP. In *ACL*, pages 264–271.
- S. Sathiya Keerthi and Chih-Jen Lin. 2003. Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural computation*, 15(7):1667–1689.
- Sandra Kübler and Eric Baucom. 2011. Fast domain adaptation for part of speech tagging for dialogues. In *RANLP*, pages 41–48.
- Percy Liang, Hal Daumé III, and Dan Klein. 2008. Structure compilation: trading structure for features. In *ICML*, pages 592–599.
- Percy Liang. 2005. Semi-supervised learning for natural language processing. Master’s thesis, Massachusetts Institute of Technology.
- Christopher D. Manning. 2011. Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In *CICLing*, pages 171–189.

- Mitchell P. 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. Reranking and self-training for parser adaptation. In *ACL*, pages 337–344.
- John Miller, Manabu Torii, and Vijay K. Shanker. 2007. Building domain-specific taggers without annotated (domain) data. In *EMNLP-CoNLL*, pages 1103–1111.
- Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *HLT-NAACL*, pages 404–411.
- Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 Shared Task on Parsing the Web. Notes of the 1st SANCL Workshop.
- Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *CoNLL*, pages 147–155.
- Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *EMNLP*, pages 133–142.
- Alexander M. Rush, Roi Reichart, Michael Collins, and Amir Globerson. 2012. Improved parsing and POS tagging using inter-sentence consistency constraints. In *EMNLP-CoNLL*, pages 1434–1444.
- Beatrice Santorini. 1990. Part-of-speech tagging guidelines for the Penn Treebank project (3rd revision, 2nd printing). Technical report, Department of Linguistics, University of Pennsylvania.
- Tobias Schnabel and Hinrich Schütze. 2013. Towards robust cross-domain domain adaptation for part-of-speech tagging. In *IJCNLP*, pages 198–206.
- Hinrich Schütze. 1993. Part-of-speech induction from scratch. In *ACL*, pages 251–258.
- Hinrich Schütze. 1995. Distributional part-of-speech tagging. In *EACL*, pages 141–148.
- Anders Søgaard. 2011. Semisupervised condensed nearest neighbor for part-of-speech tagging. In *ACL: Short papers*, pages 48–52.
- Amarnag Subramanya, Slav Petrov, and Fernando Pereira. 2010. Efficient graph-based semi-supervised learning of structured tagging models. In *EMNLP*, pages 167–176.
- Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL-HLT*, pages 173–180.
- Yoshimasa Tsuruoka and Jun’ichi Tsujii. 2005. Bidirectional inference with the easiest-first strategy for tagging sequence data. In *EMNLP-HLT*, pages 467–474.
- Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *ACL*, pages 384–394.
- Shulamit Umansky-Pesin, Roi Reichart, and Ari Rapoport. 2010. A multi-domain web-based algorithm for POS tagging of unknown words. In *COLING*, pages 1274–1282.
- Jakob Uszkoreit and Thorsten Brants. 2008. Distributed word clustering for large scale class-based language modeling in machine translation. In *ACL*, pages 755–762.

A Tabular Method for Dynamic Oracles in Transition-Based Parsing

Yoav Goldberg
Department of
Computer Science
Bar Ilan University, Israel
yoav.goldberg@gmail.com

Francesco Sartorio
Department of
Information Engineering
University of Padua, Italy
sartorio@dei.unipd.it

Giorgio Satta
Department of
Information Engineering
University of Padua, Italy
satta@dei.unipd.it

Abstract

We develop parsing oracles for two transition-based dependency parsers, including the arc-standard parser, solving a problem that was left open in (Goldberg and Nivre, 2013). We experimentally show that using these oracles during training yields superior parsing accuracies on many languages.

1 Introduction

Greedy transition-based dependency parsers (Nivre, 2008) incrementally process an input sentence from left to right. These parsers are very fast and provide competitive parsing accuracies (Nivre et al., 2007). However, greedy transition-based parsers still fall behind search-based parsers (Zhang and Clark, 2008; Huang and Sagae, 2010) with respect to accuracy.

The training of transition-based parsers relies on a component called the parsing **oracle**, which maps parser configurations to optimal transitions with respect to a gold tree. A discriminative model is then trained to simulate the oracle’s behavior. A parsing oracle is deterministic if it returns a single canonical transition. Furthermore, an oracle is partial if it is defined only for configurations that can reach the gold tree, that is, configurations representing parsing histories with no mistake. Oracles that are both deterministic and partial are called **static**. Traditionally, only static oracles have been exploited in training of transition-based parsers.

Recently, Goldberg and Nivre (2012; 2013) showed that the accuracy of greedy parsers can be substantially improved without affecting their parsing speed. This improvement relies on the introduction of novel oracles that are nondeterministic

and complete. An oracle is nondeterministic if it returns the set of all transitions that are optimal with respect to the gold tree, and it is complete if it is well-defined and correct for every configuration that is reachable by the parser. Oracles that are both nondeterministic and complete are called **dynamic**.

Goldberg and Nivre (2013) develop dynamic oracles for several transition-based parsers. The construction of these oracles is based on a property of transition-based parsers that they call arc decomposition. They also prove that the popular arc-standard system (Nivre, 2004) is not arc-decomposable, and they leave as an open research question the construction of a dynamic oracle for the arc-standard system. In this article, we develop one such oracle (§4) and prove its correctness (§5).

An extension to the arc-standard parser was presented by Sartorio et al. (2013), which relaxes the bottom-up construction order and allows mixing of bottom-up and top-down strategies. This parser, called here the LR-spine parser, achieves state-of-the-art results for greedy parsing. Like the arc-standard system, the LR-spine parser is not arc-decomposable, and a dynamic oracle for this system was not known. We extend our oracle for the arc-standard system to work for the LR-spine system as well (§6).

The dynamic oracles developed by Goldberg and Nivre (2013) for arc-decomposable systems are based on local properties of computations. In contrast, our novel dynamic oracle algorithms rely on arguably more complex structural properties of computations, which are computed through dynamic programming. This leaves open the question of whether a machine-learning model can learn to effectively simulate such complex processes: will the

benefit of training with the dynamic oracle carry over to the arc-standard and LR-spine systems? We show experimentally that this is indeed the case (§8), and that using the training-with-exploration method of (Goldberg and Nivre, 2013) with our dynamic programming based oracles yields superior parsing accuracies on many languages.

2 Arc-Standard Parser

In this section we introduce the arc-standard parser of Nivre (2004), which is the model that we use in this article. To keep the notation at a simple level, we only discuss the unlabeled version of the parser; however, a labeled extension is used in §8 for our experiments.

2.1 Preliminaries and Notation

The set of non-negative integers is denoted as \mathbb{N}_0 . For $i, j \in \mathbb{N}_0$ with $i \leq j$, we write $[i, j]$ to denote the set $\{i, i + 1, \dots, j\}$. When $i > j$, $[i, j]$ denotes the empty set.

We represent an input sentence as a string $w = w_0 \dots w_n$, $n \in \mathbb{N}_0$, where token w_0 is a special root symbol, and each w_i with $i \in [1, n]$ is a lexical token. For $i, j \in [0, n]$ with $i \leq j$, we write $w[i, j]$ to denote the substring $w_i w_{i+1} \dots w_j$ of w .

We write $i \rightarrow j$ to denote a grammatical **dependency** of some unspecified type between lexical tokens w_i and w_j , where w_i is the head and w_j is the dependent. A **dependency tree** for w is a directed, ordered tree $t = (V_w, A)$, such that $V_w = [0, n]$ is the set of nodes, $A \subseteq V_w \times V_w$ is the set of arcs, and node 0 is the root. Arc (i, j) encodes a dependency $i \rightarrow j$, and we will often use the latter notation to denote arcs.

2.2 Transition-Based Dependency Parsing

We assume the reader is familiar with the formal framework of transition-based dependency parsing originally introduced by Nivre (2003); see Nivre (2008) for an introduction. We only summarize here our notation.

Transition-based dependency parsers use a stack data structure, where each stack element is associated with a tree spanning (generating) some substring of the input w . The parser processes the input string incrementally, from left to right, applying at each step a transition that updates the stack and/or

consumes one token from the input. Transitions may also construct new dependencies, which are added to the current configuration of the parser.

We represent the **stack** data structure as an ordered sequence $\sigma = [\sigma_d, \dots, \sigma_1]$, $d \in \mathbb{N}_0$, of nodes $\sigma_i \in V_w$, with the topmost element placed at the right. When $d = 0$, we have the empty stack $\sigma = []$. Sometimes we use the vertical bar to denote the append operator for σ , and write $\sigma = \sigma' | \sigma_1$ to indicate that σ_1 is the topmost element of σ .

The parser also uses a **buffer** to store the portion of the input string still to be processed. We represent the buffer as an ordered sequence $\beta = [i, \dots, n]$ of nodes from V_w , with i the first element of the buffer. In this way β always encodes a (non-necessarily proper) suffix of w . We denote the empty buffer as $\beta = []$. Sometimes we use the vertical bar to denote the append operator for β , and write $\beta = i | \beta'$ to indicate that i is the first token of β ; consequently, we have $\beta' = [i + 1, \dots, n]$.

When processing w , the parser reaches several states, technically called configurations. A **configuration** of the parser relative to w is a triple $c = (\sigma, \beta, A)$, where σ and β are a stack and a buffer, respectively, and $A \subseteq V_w \times V_w$ is a set of arcs. The **initial** configuration for w is $([], [0, \dots, n], \emptyset)$. For the purpose of this article, a configuration is **final** if it has the form $([0], [], A)$, and in a final configuration arc set A always defines a dependency tree for w .

The core of a transition-based parser is the set of its transitions, which are specific to each family of parsers. A **transition** is a binary relation defined over the set of configurations of the parser. We use symbol \vdash to denote the union of all transition relations of a parser.

A **computation** of the parser on w is a sequence c_0, \dots, c_m , $m \in \mathbb{N}_0$, of configurations (defined relative to w) such that $c_{i-1} \vdash c_i$ for each $i \in [1, m]$. We also use the reflexive and transitive closure relation \vdash^* to represent computations. A computation is called **complete** whenever c_0 is initial and c_m is final. In this way, a complete computation is uniquely associated with a dependency tree for w .

2.3 Arc-Standard Parser

The arc-standard model uses the three types of transitions formally specified in Figure 1

$$\begin{aligned}
(\sigma, i|\beta, A) &\vdash_{\text{sh}} (\sigma|i, \beta, A) \\
(\sigma|i|j, \beta, A) &\vdash_{\text{la}} (\sigma|j, \beta, A \cup \{j \rightarrow i\}) \\
(\sigma|i|j, \beta, A) &\vdash_{\text{ra}} (\sigma|i, \beta, A \cup \{i \rightarrow j\})
\end{aligned}$$

Figure 1: Transitions in the arc-standard model.

- Shift (sh) removes the first node in the buffer and pushes it into the stack;
- Left-Arc (la) creates a new arc with the topmost node on the stack as the head and the second-topmost node as the dependent, and removes the second-topmost node from the stack;
- Right-Arc (ra) is symmetric to la in that it creates an arc with the second-topmost node as the head and the topmost node as the dependent, and removes the topmost node.

Notation We sometimes use the functional notation for a transition $\tau \in \{\text{sh}, \text{la}, \text{ra}\}$, and write $\tau(c) = c'$ in place of $c \vdash_{\tau} c'$. Naturally, sh applies only when the buffer is not empty, and la,ra require two elements on the stack. We denote by $\text{valid}(c)$ the set of valid transitions in a given configuration.

2.4 Arc Decomposition

Goldberg and Nivre (2013) show how to derive dynamic oracles for any transition-based parser which has the arc decomposition property, defined below. They also show that the arc-standard parser is not arc-decomposable.

For a configuration c , we write A_c to denote the associated set of arcs. A transition-based parser is **arc-decomposable** if, for every configuration c and for every set of arcs A that can be extended to a projective tree, we have

$$\begin{aligned}
\forall (i \rightarrow j) \in A, \exists c' [c \vdash^* c' \wedge (i \rightarrow j) \in A_{c'}] \\
\Rightarrow \exists c'' [c \vdash^* c'' \wedge A \subseteq A_{c''}].
\end{aligned}$$

In words, if each arc in A is individually derivable from c , then the set A in its entirety can be derived from c as well. The arc decomposition property is useful for deriving dynamic oracles because it is relatively easy to investigate derivability for single arcs and then, using this property, draw conclusions about the number of gold-arcs that are simultaneously derivable from the given configuration.

Unfortunately, the arc-standard parser is not arc-decomposable. To see why, consider a configuration with stack $\sigma = [i, j, k]$. Consider also arc set $A = \{(i, j), (i, k)\}$. The arc (i, j) can be derived through the transition sequence ra, ra, and the arc (i, k) can be derived through the alternative transition sequence la, ra. Yet, it is easy to see that a configuration containing both arcs cannot be reached.

As we cannot rely on the arc decomposition property, in order to derive a dynamic oracle for the arc-standard model we need to develop more sophisticated techniques which take into account the interaction among the applied transitions.

3 Configuration Loss and Dynamic Oracles

We aim to derive a dynamic oracle for the arc-standard (and related) system. This is a function that takes a configuration c and a gold tree t_G and returns a set of transitions that are “optimal” for c with respect to t_G . As already mentioned in the introduction, a dynamic oracle can be used to improve training of greedy transition-based parsers. In this section we provide a formal definition for a dynamic oracle.

Let t_1 and t_2 be two dependency trees over the same string w , with arc sets A_1 and A_2 , respectively. We define the **loss** of t_1 with respect to t_2 as

$$\mathcal{L}(t_1, t_2) = |A_1 \setminus A_2|. \quad (1)$$

Note that $\mathcal{L}(t_1, t_2) = \mathcal{L}(t_2, t_1)$, since $|A_1| = |A_2|$. Furthermore $\mathcal{L}(t_1, t_2) = 0$ if and only if t_1 and t_2 are the same tree.

Let c be a configuration of our parser relative to input string w . We write $\mathcal{D}(c)$ to denote the set of all dependency trees that can be obtained in a computation of the form $c \vdash^* c_f$, where c_f is some final configuration. We extend the loss function in (1) to configurations by letting

$$\mathcal{L}(c, t_2) = \min_{t_1 \in \mathcal{D}(c)} \mathcal{L}(t_1, t_2). \quad (2)$$

Assume some reference (desired) dependency tree t_G for w , which we call the **gold tree**. Quantity $\mathcal{L}(c, t_G)$ can be used to compute a dynamic oracle relating a parser configuration c to a set of optimal actions by setting

$$\begin{aligned}
\text{oracle}(c, t_G) = \\
\{\tau \mid \mathcal{L}(\tau(c), t_G) - \mathcal{L}(c, t_G) = 0\}. \quad (3)
\end{aligned}$$

We therefore need to develop an algorithm for computing (2). We will do this first for the arc-standard parser, and then for an extension of this model.

Notation We also apply the loss function $\mathcal{L}(t, t_G)$ in (1) when t is a dependency tree for a substring of w . In this case the nodes of t are a subset of the nodes of t_G , and $\mathcal{L}(t, t_G)$ provides a count of the nodes of t that are assigned a wrong head node, when t_G is considered as the reference tree.

4 Main Algorithm

Throughout this section we assume an arc-standard parser. Our algorithm takes as input a projective gold tree t_G and a configuration $c = (\sigma_L, \beta, A)$. We call σ_L the **left stack**, in contrast with a right stack whose construction is specified below.

4.1 Basic Idea

The algorithm consists of two steps. Informally, in the first step we compute the largest subtrees, called here tree fragments, of the gold tree t_G that have their span entirely included in the buffer β . The root nodes of these tree fragments are then arranged into a stack data structure, according to the order in which they appear in β and with the leftmost root in β being the topmost element of the stack. We call this structure the right stack σ_R . Intuitively, σ_R can be viewed as the result of pre-computing β by applying all sequences of transitions that match t_G and that can be performed independently of the stack in the input configuration c , that is, σ_L .

In the second step of the algorithm we use dynamic programming techniques to simulate all computations of the arc-standard parser starting in a configuration with stack σ_L and with a buffer consisting of σ_R , with the topmost token of σ_R being the first token of the buffer. As we will see later, the search space defined by these computations includes the dependency trees for w that are reachable from the input configuration c and that have minimum loss. We then perform a Viterbi search to pick up such value.

The second step is very similar to standard implementations of the CKY parser for context-free grammars (Hopcroft and Ullman, 1979), running on an input string obtained as the concatenation of σ_L and σ_R . The main difference is that we restrict ourselves to parse only those constituents in $\sigma_L\sigma_R$ that dominate the topmost element of σ_L (the rightmost ele-

ment, if σ_L is viewed as a string). In this way, we account for the additional constraint that we visit only those configurations of the arc-standard parser that can be reached from the input configuration c . For instance, this excludes the reduction of two nodes in σ_L that are not at the two topmost positions. This would also exclude the reduction of two nodes in σ_R : this is correct, since the associated tree fragments have been chosen as the largest such fragments in β .

The above intuitive explanation will be made mathematically precise in §5, where the notion of linear dependency tree is introduced.

4.2 Construction of the Right Stack

In the first step we process β and construct a stack σ_R , which we call the **right stack** associated with c and t_G . Each node of σ_R is the root of a tree t which satisfies the following properties

- t is a tree fragment of the gold tree t_G having span entirely included in the buffer β ;
- t is **bottom-up complete** for t_G , meaning that for each node i of t different from t 's root, the dependents of i in t_G cannot be in σ_L ;
- t is **maximal** for t_G , meaning that every super-tree of t in t_G violates the above conditions.

The stack σ_R is incrementally constructed by processing β from left to right. Each node i is copied into σ_R if it satisfies any of the following conditions

- the parent node of i in t_G is not in β ;
- some dependent of i in t_G is in σ_L or has already been inserted in σ_R .

It is not difficult to see that the nodes in σ_R are the roots of tree fragments of t_G that satisfy the condition of bottom-up completeness and the condition of maximality defined above.

4.3 Computation of Configuration Loss

We start with some notation. Let $\ell_L = |\sigma_L|$ and $\ell_R = |\sigma_R|$. We write $\sigma_L[i]$ to denote the i -th element of σ_L and $t(\sigma_L[i])$ to denote the corresponding tree fragment; $\sigma_R[i]$ and $t(\sigma_R[i])$ have a similar meaning. In order to simplify the specification of the algorithm, we assume below that $\sigma_L[1] = \sigma_R[1]$.

Algorithm 1 Computation of the loss function for the arc-standard parser

```
1:  $\mathcal{T}[1, 1](\sigma_L[1]) \leftarrow \mathcal{L}(t(\sigma_L[1]), t_G)$ 
2: for  $d \leftarrow 1$  to  $\ell_L + \ell_R - 1$  do ▷  $d$  is the index of a sub-anti-diagonal
3:   for  $j \leftarrow \max\{1, d - \ell_L + 1\}$  to  $\min\{d, \ell_R\}$  do ▷  $j$  is the column index
4:      $i \leftarrow d - j + 1$  ▷  $i$  is the row index
5:     if  $i < \ell_L$  then ▷ expand to the left
6:       for each  $h \in \Delta_{i,j}$  do
7:          $\mathcal{T}[i + 1, j](h) \leftarrow \min\{\mathcal{T}[i + 1, j](h), \mathcal{T}[i, j](h) + \delta_G(h \rightarrow \sigma_L[i + 1])\}$ 
8:          $\mathcal{T}[i + 1, j](\sigma_L[i + 1]) \leftarrow \min\{\mathcal{T}[i + 1, j](\sigma_L[i + 1]), \mathcal{T}[i, j](h) + \delta_G(\sigma_L[i + 1] \rightarrow h)\}$ 
9:       if  $j < \ell_R$  then ▷ expand to the right
10:      for each  $h \in \Delta_{i,j}$  do
11:         $\mathcal{T}[i, j + 1](h) \leftarrow \min\{\mathcal{T}[i, j + 1](h), \mathcal{T}[i, j](h) + \delta_G(h \rightarrow \sigma_R[j + 1])\}$ 
12:         $\mathcal{T}[i, j + 1](\sigma_R[j + 1]) \leftarrow \min\{\mathcal{T}[i, j + 1](\sigma_R[j + 1]), \mathcal{T}[i, j](h) + \delta_G(\sigma_R[j + 1] \rightarrow h)\}$ 
13: return  $\mathcal{T}[\ell_L, \ell_R](0) + \sum_{i \in [1, \ell_L]} \mathcal{L}(t(\sigma_L[i]), t_G)$ 
```

Therefore the elements of σ_R which have been constructed in §4.2 are $\sigma_R[i]$, $i \in [2, \ell_R]$.

Algorithm 1 uses a two-dimensional array \mathcal{T} of size $\ell_L \times \ell_R$, where each entry $\mathcal{T}[i, j]$ is an association list from integers to integers. An entry $\mathcal{T}[i, j](h)$ stores the minimum loss among dependency trees rooted at h that can be obtained by running the parser on the first i elements of stack σ_L and the first j elements of buffer σ_R . More precisely, let

$$\Delta_{i,j} = \{\sigma_L[k] \mid k \in [1, i]\} \cup \{\sigma_R[k] \mid k \in [1, j]\}. \quad (4)$$

For each $h \in \Delta_{i,j}$, the entry $\mathcal{T}[i, j](h)$ is the minimum loss among all dependency trees defined as above and with root h . We also assume that $\mathcal{T}[i, j](h)$ is initialized to $+\infty$ (not reported in the algorithm).

Algorithm 1 starts at the top-left corner of \mathcal{T} , visiting each individual sub-anti-diagonal of \mathcal{T} in ascending order, and eventually reaching the bottom-right corner of the array. For each entry $\mathcal{T}[i, j]$, the left expansion is considered (lines 5 to 8) by combining with tree fragment $\sigma_L[i + 1]$, through a left or a right arc reduction. This results in the update of $\mathcal{T}[i + 1, j](h)$, for each $h \in \Delta_{i+1,j}$, whenever a smaller value of the loss is achieved for a tree with root h . The Kronecker-like function used at line 8 provides the contribution of each single arc to the loss of the current tree. Denoting with A_G the set of

arcs of t_G , such a function is defined as

$$\delta_G(i \rightarrow j) = \begin{cases} 0, & \text{if } (i \rightarrow j) \in A_G; \\ 1, & \text{otherwise.} \end{cases} \quad (5)$$

A symmetrical process is implemented for the right expansion of $\mathcal{T}[i, j]$ through tree fragment $\sigma_R[j + 1]$ (lines 9 to 12).

As we will see in the next section, quantity $\mathcal{T}[\ell_L, \ell_R](0)$ is the minimal loss of a tree composed only by arcs that connect nodes in σ_L and σ_R . By summing the loss of all tree fragments $t(\sigma_L[i])$ to the loss in $\mathcal{T}[\ell_L, \ell_R](0)$, at line 13, we obtain the desired result, since the loss of each tree fragment $t(\sigma_R[j])$ is zero.

5 Formal Properties

Throughout this section we let w , t_G , σ_L , σ_R and $c = (\sigma_L, \beta, A)$ be defined as in §4, but we no longer assume that $\sigma_L[1] = \sigma_R[1]$. To simplify the presentation, we sometimes identify the tokens in w with the associated nodes in a dependency tree for w .

5.1 Linear Trees

Algorithm 1 explores all dependency trees that can be reached by an arc-standard parser from configuration c , under the condition that (i) the nodes in the buffer β are pre-computed into tree fragments and collapsed into their root nodes in the right stack σ_R , and (ii) nodes in σ_R cannot be combined together prior to their combination with other nodes in the left stack σ_L . This set of dependency trees is char-

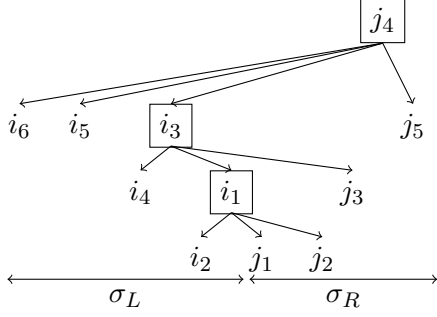


Figure 2: A possible linear tree for string pair (σ_L, σ_R) , where $\sigma_L = i_6 i_5 i_4 i_3 i_2 i_1$ and $\sigma_R = j_1 j_2 j_3 j_4 j_5$. The spine of the tree consists of nodes j_4, i_3 and i_1 .

acterized here using the notion of linear tree, to be used later in the correctness proof.

Consider two nodes $\sigma_L[i]$ and $\sigma_L[j]$ with $j > i > 1$. An arc-standard parser can construct an arc between $\sigma_L[i]$ and $\sigma_L[j]$, in any direction, only after reaching a configuration in which $\sigma_L[i]$ is at the top of the stack and $\sigma_L[j]$ is at the second topmost position. In such configuration we have that $\sigma_L[i]$ dominates $\sigma_L[1]$. Furthermore, consider nodes $\sigma_R[i]$ and $\sigma_R[j]$ with $j > i \geq 1$. Since we are assuming that tree fragments $t(\sigma_R[i])$ and $t(\sigma_R[j])$ are bottom-up complete and maximal, as defined in §4.2, we allow the construction of an arc between $\sigma_R[i]$ and $\sigma_R[j]$, in any direction, only after reaching a configuration in which $\sigma_R[i]$ dominates node $\sigma_L[1]$.

The dependency trees satisfying the restrictions above are captured by the following definition. A **linear tree** over (σ_L, σ_R) is a projective dependency tree t for string $\sigma_L \sigma_R$ satisfying both of the additional conditions reported below. The path from t 's root to node $\sigma_L[1]$ is called the **spine** of t .

- Every node of t not in the spine is a dependent of some node in the spine.
- For each arc $i \rightarrow j$ in t with j in the spine, no dependent of i can be placed in between i and j within string $\sigma_L \sigma_R$.

An example of a linear tree is depicted in Figure 2. Observe that the second condition above forbids the reduction of two nodes i and j , in case none of these dominates node $\sigma_L[1]$. For instance, the ra reduction of nodes i_3 and i_2 would result in arc $i_3 \rightarrow i_2$ replacing arc $i_1 \rightarrow i_2$ in Figure 2. The new dependency tree is not linear, because of a violation of the

second condition above. Similarly, the la reduction of nodes j_3 and j_4 would result in arc $j_4 \rightarrow j_3$ replacing arc $i_3 \rightarrow j_3$ in Figure 2, again a violation of the second condition above.

Lemma 1 Any tree $t \in \mathcal{D}(c)$ can be decomposed into trees $t(\sigma_L[i])$, $i \in [1, \ell_L]$, trees t_j , $j \in [1, q]$ and $q \geq 1$, and a linear tree t_l over $(\sigma_L, \sigma_{R,t})$, where $\sigma_{R,t} = r_1 \cdots r_q$ and each r_j is the root node of t_j . \square

PROOF (SKETCH) Trees $t(\sigma_L[i])$ are common to every tree in $\mathcal{D}(c)$, since the arc-standard model can not undo the arcs already built in the current configuration c . Similar to the construction in §4.2 of the right stack σ_R from t_G , we let t_j , $j \in [1, q]$, be tree fragments of t that cover only nodes associated with the tokens in the buffer β and that are bottom-up complete and maximal for t . These trees are indexed according to their left to right order in β . Finally, t_l is implicitly defined by all arcs of t that are not in trees $t(\sigma_L[i])$ and t_j . It is not difficult to see that t_l has a spine ending with node $\sigma_L[1]$ and is a linear tree over $(\sigma_L, \sigma_{R,t})$. \blacksquare

5.2 Correctness

Our proof of correctness for Algorithm 1 is based on a specific dependency tree t^* for w , which we define below. Let $S_L = \{\sigma_L[i] \mid i \in [1, \ell_L]\}$ and let D_L be the set of nodes that are descendants of some node in S_L . Similarly, let $S_R = \{\sigma_R[i] \mid i \in [1, \ell_R]\}$ and let D_R be the set of descendants of nodes in S_R . Note that sets S_L, S_R, D_L and D_R provide a partition of V_w .

We choose any linear tree t_l^* over (σ_L, σ_R) having root 0, such that $\mathcal{L}(t_l^*, t_G) = \min_t \mathcal{L}(t, t_G)$, where t ranges over all possible linear trees over (σ_L, σ_R) with root 0. Tree t^* consists of the set of nodes V_w and the set of arcs obtained as the union of the set of arcs of t_l^* and the set of arcs of all trees $t(\sigma_L[i])$, $i \in [1, \ell_L]$, and $t(\sigma_R[j])$, $j \in [1, \ell_R]$.

Lemma 2 $t^* \in \mathcal{D}(c)$. \square

PROOF (SKETCH) All tree fragments $t(\sigma_L[i])$ have already been parsed and are available in the stack associated with c . Each tree fragment $t(\sigma_R[j])$ can later be constructed in the computation, when a configuration c' is reached with the relevant segment of w at the start of the buffer. Note also that parsing of $t(\sigma_R[j])$ can be done in a way that does not depend on the content of the stack in c' .

Finally, the parsing of the tree fragments $t(\sigma_R[j])$ is interleaved with the construction of the arcs from the linear tree t_l^* , which are all of the form $(i \rightarrow j)$ with $i, j \in (S_L \cup S_R)$. More precisely, if $(i \rightarrow j)$ is an arc from t_l^* , at some point in the computation nodes i and j will become available at the two top-most positions in the stack. This follows from the second condition in the definition of linear tree. ■

We now show that tree t^* is “optimal” within the set $\mathcal{D}(c)$ and with respect to t_G .

Lemma 3 $\mathcal{L}(t^*, t_G) = \mathcal{L}(c, t_G)$. □

PROOF Consider an arbitrary tree $t \in \mathcal{D}(c)$. Assume the decomposition of t defined in the proof of Lemma 1, through trees $t(\sigma_L[i])$, $i \in [1, \ell_L]$, trees t_j , $j \in [1, q]$, and linear tree t_l over $(\sigma_L, \sigma_{R,t})$.

Recall that an arc $i \rightarrow j$ denotes an ordered pair (i, j) . Let us consider the following partition for the set of arcs of any dependency tree for w

$$\begin{aligned} A_1 &= (S_L \cup D_L) \times D_L, \\ A_2 &= (S_R \cup D_R) \times D_R, \\ A_3 &= (V_w \times V_w) \setminus (A_1 \cup A_2). \end{aligned}$$

In what follows, we compare the losses $\mathcal{L}(t, t_G)$ and $\mathcal{L}(t^*, t_G)$ by separately looking into the contribution to such quantities due to the arcs in A_1 , A_2 and A_3 .

Note that the arcs of trees $t(\sigma_L[i])$ are all in A_1 , the arcs of trees $t(\sigma_R[j])$ are all in A_2 , and the arcs of tree t_l^* are all in A_3 . Since t and t^* share trees $t(\sigma_L[i])$, when restricted to arcs in A_1 quantities $\mathcal{L}(t, t_G)$ and $\mathcal{L}(t^*, t_G)$ are the same. When restricted to arcs in A_2 , quantity $\mathcal{L}(t^*, t_G)$ is zero, by construction of the trees $t(\sigma_R[j])$. Thus $\mathcal{L}(t, t_G)$ can not be smaller than $\mathcal{L}(t^*, t_G)$ for these arcs. The difficult part is the comparison of the contribution to $\mathcal{L}(t, t_G)$ and $\mathcal{L}(t^*, t_G)$ due to the arcs in A_3 . We deal with this below.

Let $A_{S,G}$ be the set of all arcs from t_G that are also in set $(S_L \times S_R) \cup (S_R \times S_L)$. In words, $A_{S,G}$ represents gold arcs connecting nodes in S_L and nodes in S_R , in any direction. Within tree t , these arcs can only be found in the t_l component, since nodes in S_L are all placed within the spine of t_l , or else at the left of that spine.

Let us consider an arc $(j \rightarrow i) \in A_{S,G}$ with $j \in S_L$ and $i \in S_R$, and let us assume that $(j \rightarrow i)$ is in t_l^* . If token a_i does not occur in $\sigma_{R,t}$, node i is not

in t_l and $(j \rightarrow i)$ can not be an arc of t . We then have that $(j \rightarrow i)$ contributes one unit to $\mathcal{L}(t, t_G)$ but does not contribute to $\mathcal{L}(t^*, t_G)$. Similarly, let $(i \rightarrow j) \in A_{S,G}$ be such that $i \in S_R$ and $j \in S_L$, and assume that $(i \rightarrow j)$ is in t_l^* . If token a_i does not occur in $\sigma_{R,t}$, arc $(i \rightarrow j)$ can not be in t . We then have that $(i \rightarrow j)$ contributes one unit to $\mathcal{L}(t, t_G)$ but does not contribute to $\mathcal{L}(t^*, t_G)$.

Intuitively, the above observations mean that the winning strategy for trees in $\mathcal{D}(c)$ is to move nodes from S_R as much as possible into the linear tree component t_l , in order to make it possible for these nodes to connect to nodes in S_L , in any direction. In this case, arcs from A_3 will also move into the linear tree component of a tree in $\mathcal{D}(c)$, as it happens in the case of t^* . We thus conclude that, when restricted to the set of arcs in A_3 , quantity $\mathcal{L}(t, t_G)$ is not smaller than $\mathcal{L}(t^*, t_G)$, because stack σ_R has at least as many tokens corresponding to nodes in S_R as stack $\sigma_{R,t}$, and because t_l^* has the minimum loss among all the linear trees over (σ_L, σ_R) .

Putting all of the above observations together, we conclude that $\mathcal{L}(t, t_G)$ can not be smaller than $\mathcal{L}(t^*, t_G)$. This concludes the proof, since t has been arbitrarily chosen in $\mathcal{D}(c)$. ■

Theorem 1 Algorithm 1 computes $\mathcal{L}(c, t_G)$. □

PROOF (SKETCH) Algorithm 1 implements a Viterbi search for trees with smallest loss among all linear trees over (σ_L, σ_R) . Thus $\mathcal{T}[\ell_L, \ell_R](0) = \mathcal{L}(t_l^*, t_G)$. The loss of the tree fragments $t(\sigma_R[j])$ is 0 and the loss of the tree fragments $t(\sigma_L[i])$ is added at line 13 in the algorithm. Thus the algorithm returns $\mathcal{L}(t^*, t_G)$, and the statement follows from Lemma 2 and Lemma 3. ■

5.3 Computational Analysis

Following §4.2, the right stack σ_R can be easily constructed in time $\mathcal{O}(n)$, n the length of the input string. We now analyze Algorithm 1. For each entry $\mathcal{T}[i, j]$ and for each $h \in \Delta_{i,j}$, we update $\mathcal{T}[i, j](h)$ a number of times bounded by a constant which does not depend on the input. Each updating can be computed in constant time as well. We thus conclude that Algorithm 1 runs in time $\mathcal{O}(\ell_L \cdot \ell_R \cdot (\ell_L + \ell_R))$. Quantity $\ell_L + \ell_R$ is bounded by n , but in practice the former is significantly smaller. When measured over the sentences in the Penn

Treebank, the average value of $\frac{\ell_L + \ell_R}{n}$ is 0.29. In terms of runtime, training is 2.3 times slower when using our oracle instead of a static oracle.

6 Extension to the LR-Spine Parser

In this section we consider the transition-based parser proposed by Sartorio et al. (2013), called here the LR-spine parser. This parser is not arc-decomposable: the same example reported in §2.4 can be used to show this fact. We therefore extend to the LR-spine parser the algorithm developed in §4.

6.1 The LR-Spine Parser

Let t be a dependency tree. The **left spine** of t is an ordered sequence $\langle i_1, \dots, i_p \rangle$, $p \geq 1$, consisting of all nodes in a descending path from the root of t taking the leftmost child node at each step. The **right spine** of t is defined symmetrically. We use \oplus to denote sequence concatenation.

In the LR-spine parser each stack element $\sigma[i]$ denotes a partially built subtree $t(\sigma[i])$ and is represented by a pair (ls_i, rs_i) , with ls_i and rs_i the left and the right spine, respectively, of $t(\sigma[i])$. We write $ls_i[k]$ ($rs_i[k]$) to represent the k -th element of ls_i (rs_i , respectively). We also write $r(\sigma[i])$ to denote the root of $t(\sigma[i])$, so that $r(\sigma[i]) = ls_i[1] = rs_i[1]$.

Informally, the LR-spine parser uses the same transition typologies as the arc-standard parser. However, an arc ($h \rightarrow d$) can now be created with the head node h chosen from any node in the spine of the involved tree. The transition types of the LR-spine parser are defined as follows.

- Shift (sh) removes the first node from the buffer and pushes into the stack a new element, consisting of the left and right spines of the associated tree

$$(\sigma, i | \beta, A) \vdash_{\text{sh}} (\sigma | (\langle i \rangle, \langle i \rangle), \beta, A) .$$

- Left-Arc k (la_k) creates a new arc $h \rightarrow d$ from the k -th node in the left spine of the topmost tree in the stack to the head of the second element in the stack. Furthermore, the two topmost stack elements are replaced by a new element associated with the resulting tree

$$(\sigma' | \sigma[2] | \sigma[1], \beta, A) \vdash_{la_k} (\sigma' | \sigma_{la_k}, \beta, A \cup \{h \rightarrow d\})$$

where we have set $h = ls_1[k]$, $d = r(\sigma[2])$ and $\sigma_{la_k} = (\langle ls_1[1], \dots, ls_1[k] \rangle \oplus ls_2, rs_1)$.

- Right-Arc k (ra_k for short) is defined symmetrically with respect to la_k

$$(\sigma' | \sigma[2] | \sigma[1], \beta, A) \vdash_{ra_k} (\sigma' | \sigma_{ra_k}, \beta, A \cup \{h \rightarrow d\})$$

where we have set $h = rs_2[k]$, $d = r(\sigma[1])$ and $\sigma_{ra_k} = (ls_2, \langle rs_2[1], \dots, rs_2[k] \rangle \oplus rs_1)$.

Note that, at each configuration in the LR-spine parser, there are $|ls_1|$ possible la_k transitions, one for each choice of a node in the left spine of $t(\sigma[1])$; similarly, there are $|rs_2|$ possible ra_k transitions, one for each choice of a node in the right spine of $t(\sigma[2])$.

6.2 Configuration Loss

We only provide an informal description of the extended algorithm here, since it is very similar to the algorithm in §4.

In the first phase we use the procedure of §4.2 for the construction of the right stack σ_R , considering only the roots of elements in σ_L and ignoring the rest of the spines. The only difference is that each element $\sigma_R[j]$ is now a pair of spines $(ls_{R,j}, rs_{R,j})$. Since tree fragment $t(\sigma_R[j])$ is bottom-up complete (see §4.1), we now restrict the search space in such a way that only the root node $r(\sigma_R[j])$ can take dependents. This is done by setting $ls_{R,j} = rs_{R,j} = \langle r(\sigma_R[j]) \rangle$ for each $j \in [1, \ell_R]$. In order to simplify the presentation we also assume $\sigma_R[1] = \sigma_L[1]$, as done in §4.3.

In the second phase we compute the loss of an input configuration using a two-dimensional array \mathcal{T} , defined as in §4.3. However, because of the way transitions are defined in the LR-spine parser, we now need to distinguish tree fragments not only on the basis of their roots, but also on the basis of their left and right spines. Accordingly, we define each entry $\mathcal{T}[i, j]$ as an association list with keys of the form (ls, rs) . More specifically, $\mathcal{T}[i, j](ls, rs)$ is the minimum loss of a tree with left and right spines ls and rs , respectively, that can be obtained by running the parser on the first i elements of stack σ_L and the first j elements of buffer σ_R .

We follow the main idea of Algorithm 1 and expand each tree in $\mathcal{T}[i, j]$ at its left side, by combining with tree fragment $t(\sigma_L[i + 1])$, and at its right side, by combining with tree fragment $t(\sigma_R[j + 1])$.

Tree combination deserves some more detailed discussion, reported below.

We consider the combination of a tree t_a from $\mathcal{T}[i, j]$ and tree $t(\sigma_L[i + 1])$ by means of a left-arc transition. All other cases are treated symmetrically. Let (ls_a, rs_a) be the spine pair of t_a , so that the loss of t_a is stored in $\mathcal{T}[i, j](ls_a, rs_a)$. Let also (ls_b, rs_b) be the spine pair of $t(\sigma_L[i + 1])$. In case there exists a gold arc in t_G connecting a node from ls_a to $r(\sigma_L[i + 1])$, we choose the transition la_k , $k \in [1, |ls_a|]$, that creates such arc. In case such gold arc does not exist, we choose the transition la_k with the maximum possible value of k , that is, $k = |ls_a|$. We therefore explore only one of the several possible ways of combining these two trees by means of a left-arc transition.

We remark that the above strategy is safe. In fact, in case the gold arc exists, no other gold arc can ever involve the nodes of ls_a eliminated by la_k (see the definition in §6.1), because arcs can not cross each other. In case the gold arc does not exist, our choice of $k = |ls_a|$ guarantees that we do not eliminate any element from ls_a .

Once a transition la_k is chosen, as described above, the reduction is performed and the spine pair (ls, rs) for the resulting tree is computed from (ls_a, rs_a) and (ls_b, rs_b) , as defined in §6.1. At the same time, the loss of the resulting tree is computed, on the basis of the loss $\mathcal{T}[i, j](ls_a, rs_a)$, the loss of tree $t(\sigma_L[i + 1])$, and a Kronecker-like function defined below. This loss is then used to update $\mathcal{T}[i + 1, j](ls, rs)$.

Let t_a and t_b be two trees that must be combined in such a way that t_b becomes the dependent of some node in one of the two spines of t_a . Let also $p_a = (ls_a, rs_a)$ and $p_b = (ls_b, rs_b)$ be spine pairs for t_a and t_b , respectively. Recall that A_G is the set of arcs of t_G . The new Kronecker-like function for the computation of the loss is defined as

$$\delta_G(p_a, p_b) = \begin{cases} 0, & \text{if } r(t_a) < r(t_b) \wedge \\ & \exists k[(rs_a^k \rightarrow r(t_b)) \in A_G]; \\ 0, & \text{if } r(t_a) > r(t_b) \wedge \\ & \exists k[(ls_a^k \rightarrow r(t_b)) \in A_G]; \\ 1, & \text{otherwise.} \end{cases}$$

6.3 Efficiency Improvement

The algorithm in §6.2 has an exponential behaviour. To see why, consider trees in $\mathcal{T}[i, j]$. These trees are produced by the combination of trees in $\mathcal{T}[i - 1, j]$ with tree $t(\sigma_L[i])$, or by the combination of trees in $\mathcal{T}[i, j - 1]$ with tree $t(\sigma_R[j])$. Since each combination involves either a left-arc or a right-arc transition, we obtain a recursive relation that resolves into a number of trees in $\mathcal{T}[i, j]$ bounded by 4^{i+j-2} .

We introduce now two restrictions to the search space of our extended algorithm that result in a huge computational saving. For a spine s , we write $\mathcal{N}(s)$ to denote the set of all nodes in s . We also let $\Delta_{i,j}$ be the set of all pairs (ls, rs) such that $\mathcal{T}[i, j](ls, rs) \neq +\infty$.

- Every time a new pair (ls, rs) is created in $\Delta_{i,j}$, we remove from ls all nodes different from the root that do not have gold dependents in $\{r(\sigma_L[k]) \mid k < i\}$, and we remove from rs all nodes different from the root that do not have gold dependents in $\{r(\sigma_R[k]) \mid k > j\}$.
- A pair $p_a = (ls_a, rs_a)$ is removed from $\Delta_{i,j}$ if there exists a pair $p_b = (ls_b, rs_b)$ in $\Delta_{i,j}$ with the same root node as p_a and with $(ls_a, rs_a) \neq (ls_b, rs_b)$, such that $\mathcal{N}(ls_a) \subseteq \mathcal{N}(ls_b)$, $\mathcal{N}(rs_a) \subseteq \mathcal{N}(rs_b)$, and $\mathcal{T}[i, j](p_a) \geq \mathcal{T}[i, j](p_b)$.

The first restriction above reduces the size of a spine by eliminating a node if it is irrelevant for the computation of the loss of the associated tree. The second restriction eliminates a tree t_a if there is a tree t_b with smaller loss than t_a , such that in the computations of the parser t_b provides exactly the same context as t_a . It is not difficult to see that the above restrictions do not affect the correctness of the algorithm, since they always leave in our search space some tree that has optimal loss.

A mathematical analysis of the computational complexity of the extended algorithm is quite involved. In Figure 3, we plot the worst case size of $\mathcal{T}[i, j]$ for each value of $j + i - 1$, computed over all configurations visited in the training phase (see §7). We see that $|\mathcal{T}[i, j]|$ grows linearly with $j + i - 1$, leading to the same space requirements of Algorithm 1. Empirically, training with the dynamic

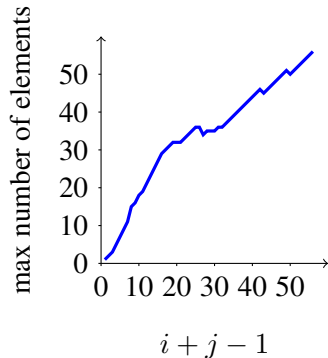


Figure 3: Empirical worst case size of $\mathcal{T}[i, j]$ for each value of $i + j - 1$ as measured on the Penn Treebank corpus.

Algorithm 2 Online training for greedy transition-based parsers

```

1:  $\mathbf{w} \leftarrow 0$ 
2: for  $k$  iterations do
3:   shuffle(corpus)
4:   for sentence  $w$  and gold tree  $t_G$  in corpus do
5:      $c \leftarrow \text{INITIAL}(w)$ 
6:     while not FINAL( $c$ ) do
7:        $\tau_p \leftarrow \text{argmax}_{\tau \in \text{valid}(c)} \mathbf{w} \cdot \phi(c, \tau)$ 
8:        $\tau_o \leftarrow \text{argmax}_{\tau \in \text{oracle}(c, t_G)} \mathbf{w} \cdot \phi(c, \tau)$ 
9:       if  $\tau_p \notin \text{oracle}(c, t_G)$  then
10:         $\mathbf{w} \leftarrow \mathbf{w} + \phi(c, \tau_o) - \phi(c, \tau_p)$ 
11:         $\tau \leftarrow \begin{cases} \tau_p & \text{if EXPLORE} \\ \tau_o & \text{otherwise} \end{cases}$ 
12:         $c \leftarrow \tau(c)$ 
return averaged( $\mathbf{w}$ )

```

oracle is only about 8 times slower than training with the oracle of Sartorio et al. (2013) without exploring incorrect configurations.

7 Training

We follow the training procedure suggested by Goldberg and Nivre (2013), as described in Algorithm 2. The algorithm performs online learning using the averaged perceptron algorithm. A weight vector \mathbf{w} (initialized to 0) is used to score the valid transitions in each configuration based on a feature representation ϕ , and the highest scoring transition τ_p is predicted. If the predicted transition is not optimal according to the oracle, the weights \mathbf{w} are updated away from the predicted transition and to-

wards the highest scoring oracle transition τ_o . The parser then moves to the next configuration, by taking either the predicted or the oracle transition. In the “error exploration” mode (EXPLORE is true), the parser follows the predicted transition, and otherwise the parser follows the oracle transition. Note that the error exploration mode requires the completeness property of a dynamic oracle.

We consider three training conditions: *static*, in which the oracle is deterministic (returning a single canonical transition for each configuration) and no error exploration is performed; *nondet*, in which we use a nondeterministic partial oracle (Sartorio et al., 2013), but do not perform error exploration; and *explore* in which we use the dynamic oracle and perform error exploration. The *static* setup mirrors the way greedy parsers are traditionally trained. The *nondet* setup allows the training procedure to choose which transition to take in case of spurious ambiguities. The *explore* setup increases the configuration space explored by the parser during training, by exposing the training procedure to non-optimal configurations that are likely to occur during parsing, together with the optimal transitions to take in these configurations. It was shown by Goldberg and Nivre (2012; 2013) that the *nondet* setup outperforms the *static* setup, and that the *explore* setup outperforms the *nondet* setup.

8 Experimental Evaluation

Datasets Performance evaluation is carried out on CoNLL 2007 multilingual dataset, as well as on the Penn Treebank (PTB) (Marcus et al., 1993) converted to Stanford basic dependencies (De Marneffe et al., 2006). For the CoNLL datasets we use gold part-of-speech tags, while for the PTB we use automatically assigned tags. As usual, the PTB parser is trained on sections 2-21 and tested on section 23.

Setup We train labeled versions of the arc-standard (std) and LR-spine (lrs) parsers under the *static*, *nondet* and *explore* setups, as defined in §7. In the *nondet* setup we use a nondeterministic partial oracle and in the *explore* setup we use the nondeterministic complete oracles we present in this paper. In the *static* setup we resolve oracle ambiguities and choose a canonic transition sequence by attaching arcs as soon as possible. In the *explore* setup,

parser:train	Arabic	Basque	Catalan	Chinese	Czech	English	Greek	Hungarian	Italian	Turkish	PTB
	UAS										
std:static	81.39	75.37	90.32	85.17	78.90	85.69	79.90	77.67	82.98	77.04	89.86
std:nondet	81.33	74.82	90.75	84.80	79.92	86.89	81.19	77.51	84.15	76.85	90.56
std:explore	82.56	74.39	90.95	85.65	81.01	87.70	81.85	78.72	84.37	77.21	90.92
lrs:static	81.67	76.07	91.47	84.24	77.93	86.36	79.43	76.56	84.64	77.00	90.33
lrs:nondet	83.14	75.53	91.31	84.98	80.03	88.38	81.12	76.98	85.29	77.63	91.18
lrs:explore	84.54	75.82	91.92	86.72	81.19	89.37	81.78	77.48	85.38	78.61	91.77
	LAS										
std:static	71.93	65.64	84.90	80.35	71.39	84.60	72.25	67.66	78.77	65.90	87.56
std:nondet	71.09	65.28	85.36	80.06	71.47	85.91	73.40	67.77	80.06	65.81	88.30
std:explore	72.89	65.27	85.82	81.28	72.92	86.79	74.22	69.57	80.25	66.71	88.72
lrs:static	72.24	66.21	86.02	79.36	70.48	85.38	72.36	66.79	80.38	66.02	88.07
lrs:nondet	72.94	65.66	86.03	80.47	71.32	87.45	73.09	67.70	81.32	67.02	88.96
lrs:explore	74.54	66.91	86.83	82.38	72.72	88.44	74.04	68.76	81.50	68.06	89.53

Table 1: Scores on the CoNLL 2007 dataset (including punctuation, gold parts of speech) and on Penn Tree Bank (excluding punctuation, predicted parts of speech). Label ‘std’ refers to the arc-standard parser, and ‘lrs’ refers to the LR-spine parser. Each number is an average over 5 runs with different randomization seeds.

from the first round of training onward, we always follow the predicted transition (EXPLORE is true). For all languages, we deal with non-projectivity by skipping the non-projective sentences during training but not during test. For each parsing system, we use the same feature templates across all languages.¹ The arc-standard models are trained for 15 iterations and the LR-spine models for 30 iterations, after which all the models seem to have converged.

Results In Table 1 we report the labeled (LAS) and unlabeled (UAS) attachment scores. As expected, the LR-spine parsers outperform the arc-standard parsers trained under the same setup. Training with the dynamic oracles is also beneficial: despite the arguable complexity of our proposed oracles, the trends are consistent with those reported by Goldberg and Nivre (2012; 2013). For the arc-standard model we observe that the move from a static to a nondeterministic oracle during training improves the accuracy for most of languages. Making use of the completeness of the dynamic oracle and moving to the error exploring setup further improve results. The only exceptions are Basque, that has a small dataset with more than 20% of non-projective sentences, and Chinese. For Chinese we observe a reduction of accuracy in the *nondet* setup, but an increase in the *explore* setup.

For the LR-spine parser we observe a practically constant increase of performance by moving from

¹Our complete code, together with the description of the feature templates, is available on the second author’s homepage.

the static to the nondeterministic and then to the error exploring setups.

9 Conclusions

We presented dynamic oracles, based on dynamic programming, for the arc-standard and the LR-spine parsers. Empirical evaluation on 10 languages showed that, despite the apparent complexity of the oracle calculation procedure, the oracles are still learnable, in the sense that using these oracles in the error exploration training algorithm presented in (Goldberg and Nivre, 2012) considerably improves the accuracy of the trained parsers.

Our algorithm computes a dynamic oracle using dynamic programming to explore a forest of dependency trees that can be reached from a given parser configuration. For the arc-standard parser, the computation takes cubic time in the size of the largest of the left and right input stacks. Dynamic programming for the simulation of arc-standard parsers have been proposed by Kuhlmann et al. (2011). That algorithm could be adapted to compute minimum loss for a given configuration, but the running time is $\mathcal{O}(n^4)$, n the size of the input string: besides being asymptotically slower by one order of magnitude, in practice n is also larger than the stack size above.

Acknowledgments We wish to thank the anonymous reviewers. In particular, we are indebted to one of them for two important technical remarks. The third author has been partially supported by MIUR under project PRIN No. 2010LYA9RH.006.

References

- Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, volume 6, pages 449–454.
- Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proc. of the 24th COLING*, Mumbai, India.
- Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *Transactions of the association for Computational Linguistics*, 1.
- John E. Hopcroft and Jeffrey D. Ullman. 1979. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, Reading, MA.
- Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, July.
- Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic programming algorithms for transition-based dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 673–682, Portland, Oregon, USA, June. Association for Computational Linguistics.
- Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.
- Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of EMNLP-CoNLL*.
- Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the Eighth International Workshop on Parsing Technologies (IWPT)*, pages 149–160, Nancy, France.
- Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57, Barcelona, Spain.
- Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.
- Francesco Sartorio, Giorgio Satta, and Joakim Nivre. 2013. A transition-based dependency parser using a dynamic parsing strategy. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 135–144, Sofia, Bulgaria, August. Association for Computational Linguistics.
- Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of EMNLP*.

Temporal Annotation in the Clinical Domain

William F. Styler IV¹, Steven Bethard², Sean Finan³, Martha Palmer¹,
Sameer Pradhan³, Piet C de Groen⁴, Brad Erickson⁴, Timothy Miller³,
Chen Lin³, Guergana Savova³ and James Pustejovsky⁵

¹ Department of Linguistics, University of Colorado at Boulder

² Department of Computer and Information Sciences, University of Alabama at Birmingham

³ Children's Hospital Boston Informatics Program and Harvard Medical School

⁴ Mayo Clinic College of Medicine, Mayo Clinic, Rochester, MN

⁵ Department of Computer Science, Brandeis University

Abstract

This article discusses the requirements of a formal specification for the annotation of temporal information in clinical narratives. We discuss the implementation and extension of ISO-TimeML for annotating a corpus of clinical notes, known as the THYME corpus. To reflect the information task and the heavily inference-based reasoning demands in the domain, a new annotation guideline has been developed, “the THYME Guidelines to ISO-TimeML (THYME-TimeML)”. To clarify what relations merit annotation, we distinguish between linguistically-derived and inferentially-derived temporal orderings in the text. We also apply a top performing TempEval 2013 system against this new resource to measure the difficulty of adapting systems to the clinical domain. The corpus is available to the community and has been proposed for use in a SemEval 2015 task.

1 Introduction

There is a long-standing interest in temporal reasoning within the biomedical community (Savova et al., 2009; Hripcsak et al., 2009; Meystre et al., 2008; Bramsen et al., 2006; Combi et al., 1997; Keravnou, 1997; Dolin, 1995; Irvine et al., 2008; Sullivan et al., 2008). This interest extends to the automatic extraction and interpretation of temporal information from medical texts, such as electronic discharge summaries and patient case summaries. Making effective use of temporal information from such narratives is a crucial step in the intelligent analysis of informatics for medical researchers, while an awareness of temporal information (both implicit and explicit) in a text is also necessary for many data mining tasks.

It has also been demonstrated that the temporal information in clinical narratives can be usefully mined

to provide information for some higher-level temporal reasoning (Zhao et al., 2005). Robust temporal understanding of such narratives, however, has been difficult to achieve, due to the complexity of determining temporal relations among events, the diversity of temporal expressions, and the interaction with broader computational linguistic issues.

Recent work on Electronic Health Records (EHRs) points to new ways to exploit and mine the information contained therein (Savova et al., 2009; Roberts et al., 2009; Zheng et al., 2011; Turchin et al., 2009). We target two main use cases for extracted data. First, we hope to enable interactive displays and summaries of the patient's records to the physician at the time of visit, making a comprehensive review of the patient's history both faster and less prone to oversights. Second, we hope to enable temporally-aware secondary research across large databases of medical records (e.g., “What percentage of patients who undergo procedure X develop side-effect Y within Z months?”). Both of these applications require the extraction of time and date associations for critical events and the relative ordering of events during the patient's period of care, all from the various records which make up a patient's EHR. Although we have these two specific applications in mind, the schema we have developed is generalizable and could potentially be embedded in a wide variety of biomedical use cases.

Narrative texts in EHRs are temporally rich documents that frequently contain assertions about the timing of medical events, such as visits, laboratory values, symptoms, signs, diagnoses, and procedures (Bramsen et al., 2006; Hripcsak et al., 2009; Zhou et al., 2008). Temporal representation and reasoning in the medical record are difficult due to: (1) the diversity of time expressions; (2) the complexity of determining temporal relations among events (which are often left to inference); (3) the difficulty of handling the temporal granularity of an event; and (4)

general issues in natural language processing (e.g., ambiguity, anaphora, ellipsis, conjunction). As a result, the signals used for reconstructing a timeline can be both domain-specific and complex, and are often left implicit, requiring significant domain knowledge to accurately detect and interpret.

In this paper, we discuss the demands on accurately annotating such temporal information in clinical notes. We describe an implementation and extension of ISO-TimeML (Pustejovsky et al., 2010), developed specifically for the clinical domain, which we refer to as the “THYME Guidelines to ISO-TimeML” (“THYME-TimeML”), where THYME stands for “Temporal Histories of Your Medical Events”. A simplified version of these guidelines formed the basis for the 2012 i2b2 medical-domain temporal relation challenge (Sun et al., 2013a).

This is being developed in the context of the THYME project, whose goal is to both create robust gold standards for semantic information in clinical notes, as well as to develop state-of-the-art algorithms to train and test on this dataset.

Deriving timelines from news text requires the concrete realization of context-dependent assumptions about temporal intervals, orderings and organization, underlying the explicit signals marked in the text (Pustejovsky and Stubbs, 2011). Deriving patient history timelines from clinical notes also involves these types of assumptions, but there are special demands imposed by the characteristics of the clinical narrative. Due to both medical shorthand practices and general domain knowledge, many event-event relations are not signaled in the text at all, and rely on a shared understanding and common conceptual models of the progressions of medical procedures available only to readers familiar with language use in the medical community.

Identifying these implicit relations and temporal properties puts a heavy burden on the annotation process. As such, in the THYME-TimeML guideline, considerable effort has gone into both describing and proscribing the annotation of temporal orderings that are inferable only through domain-specific temporal knowledge.

Although the THYME guidelines describe a number of departures from the ISO-TimeML standard for expediency and ease of annotation, this paper will focus on those differences specifically motivated by the needs of the clinical domain, and on the consequences for systems built to extract temporal data in

both the clinical and general domain.

2 The Nature of Clinical Documents

In the THYME corpus, we have been examining 1,254 de-identified¹ notes from a large healthcare practice (the Mayo Clinic), representing two distinct fields within oncology: brain cancer, and colon cancer. To date, we have principally examined two different general types of clinical narrative in our EHRs: clinical notes and pathology reports.

Clinical notes are records of physician interactions with a patient, and often include multiple, clearly delineated sections detailing different aspects of the patient’s care and present illness. These notes are fairly generic across institutions and specialities, and although some terms and inferences may be specific to a particular type of practice (such as oncology), they share a uniform structure and pattern. The ‘History of Present Illness’, for example, summarizes the course of the patient’s chief complaint, as well as the interventions and diagnostics which have been thus far attempted. In other sections, the doctor may outline her current plan for the patient’s treatment, then later describe the patient’s specific medical history, allergies, care directives, and so forth.

Most critically for temporal reasoning, each clinical note reflects a single time in the patient’s treatment history at which all of the doctor’s statements are accurate (the DOCTIME), and each section tends to describe events of a particular timeframe. For example, ‘History of Present illness’ predominantly describes events occurring before DOCTIME, whereas ‘Medications’ provides a snapshot at DOCTIME and ‘Ongoing Care Orders’ discusses events which have not yet occurred.²

Clinical notes contain rich temporal information and background, moving fluidly from prior treatments and symptoms to present conditions to future interventions. They are also often rich with hypothetical statements (“if the tumor recurs, we can...”), each of which can form its own separate timeline.

By contrast, pathology notes are quite different. Such notes are generated by a medical pathologist

¹Although most patient information was removed, dates and temporal information were not modified according to this project’s specific data use agreement.

²One complication is the propensity of doctors and automated systems to later update sections in a note without changing the timestamp or metadata. We have added a SECTIONTIME to keep these updated sections from affecting our overall timeline.

upon receipt and analysis of specimens (ranging from tissue samples from biopsy to excised portions of tumor or organs). Pathology notes provide crucial information to the patient’s doctor confirming the malignancy (cancer) in samples, describing surgical margins (which indicate whether a tumor was completely excised), and classifying and ‘staging’ a tumor, describing the severity and spread of the cancer. Because the information in such notes pertains to samples taken at a single moment in time, they are temporally sparse, seldom referring to events before or after the examination of the specimen. However, they contain critical information about the state of the patient’s illness and about the cancer itself, and must be interpreted to understand the history of the patient’s illness.

Most importantly, in all EHRs, we must contend with the results of a fundamental tension in modern medical records: hyper-detailed records provide a crucial defense against malpractice litigation, but including such detail takes enormous time, which doctors seldom have. Given that these notes are written by and for medical professionals (who form a relatively insular speech community), a great many non-standard expressions, abbreviations, and assumptions of shared knowledge are used, which are simultaneously concise and detail-rich for others who have similar backgrounds.

These time-saving devices can range from temporally loaded acronyms (e.g., ‘qid’, Latin for *quater in die*, ‘four times daily’), to assumed orderings (a diagnostic test for a disorder is assumed to come before the procedure which treats it), and even to completely implicit events and temporal details. For example, consider the sentence in (1).

(1) **Colonoscopy** 3/12/10, nodule **biopsies** negative
We must understand that during the colonoscopy, the doctor obtained biopsies of nodules, which were packaged and sent to a pathologist, who reviewed them and determined them to be ‘negative’ (non-cancerous).

In such documents, we must recover as much temporal detail as possible, even though it may be expressed in a way which is not easily understood outside of the medical community, let alone by linguists or automated systems. We must also be aware of the legal relevance of some events (e.g., “**We discussed** the possible side effects”), even when they may not seem relevant to the patient’s actual care.

Finally, each specialty and note type has separate

conventions. Within colon cancer notes, the American Joint Committee on Cancer (AJCC) Staging Codes (e.g., *T4N1*, indicating the nature of the tumor, lymph node and metastasis involvement) are meticulously recorded, but are largely absent in the brain cancer notes which make up the second corpus in our project. So, although clinical notes share many similarities, annotators without sufficient domain expertise may require additional training to adapt to the inferences and nuances of a new clinical subdomain.

3 Interpreting ‘Event’ and Temporal Expressions in the Clinical Domain

Much prior work has been done on standardizing the annotation of events and temporal expressions in text. The most widely used approach is the ISO-TimeML specification (Pustejovsky et al., 2010), an ISO standard that provides a common framework for annotating and analyzing time, events, and event relations. As defined by ISO-TimeML, an EVENT refers to anything that can be said “to obtain or hold true, to happen or to occur”. This is a broad notion of event, consistent with Bach’s use of the term “eventuality” (Bach, 1986) as well as the notion of fluents in AI (McCarthy, 2002).

Because the goals of the THYME project involve automatically identifying the clinical timeline for a patient from clinical records, the scope of what should be admitted into the domain of events is interpreted more broadly than in ISO-TimeML³. Within the THYME-TimeML guideline, an EVENT is *anything* relevant to the clinical timeline, i.e., anything that would show up on a detailed timeline of the patient’s care or life. The best single-word syntactic head for the EVENT is then used as its span. For example, a *diagnosis* would certainly appear on such a timeline, as would a *tumor*, *illness*, or *procedure*. On the other hand, entities that persist throughout the relevant temporal period of the clinical timeline (*endurants* in ontological circles) would not be considered as event-like. This includes the patient, other humans mentioned (the patient’s mother-in-law or the doctor), organizations (the emergency room), non-anatomical objects (the patient’s car), or individual parts of the patient’s anatomy (an arm is not an EVENT unless missing or otherwise notable).

To meet our explicit goals, the THYME-TimeML guideline introduces two additional levels of interpre-

³Our use of the term ‘EVENT’ corresponds with the less specific ISO-TimeML term ‘Eventuality’

tation beyond that specified by ISO-TimeML: (i) a well-defined task; and (ii) a clearly identified domain. By focusing on the creation of a *clinical timeline* from *clinical narrative*, the guideline imposes constraints that cannot be assumed for a broadly defined and domain independent annotation schema.

Some EVENTS annotated under our guideline are considered meaningful and eventive mostly by virtue of a specific clinical or legal value. For example, AJCC Staging Codes (discussed in Section 2) are eventive only in the sense of the code being assigned to a tumor at a given moment in the patient's care. However, they are of such critical importance and informative value to doctors that we have chosen to annotate them *specifically so that they will show up on the patient's timeline in a clinical setting*.

Similarly, because of legal pressures to establish informed consent and patient knowledge of risk, entire paragraphs of clinical notes are dedicated to documenting the doctor's discussion of risks, plans, and alternative strategies. As such, we annotate verbs of discussion ("We **talked** about the risks of this drug"), consent ("She **agreed** with the current plan"), and comprehension ("Mrs. Larsen **repeated** the potential side effects back to me"), even though they are more relevant to legal defense than medical treatment.

It is also because of this grounding in clinical language that entities and other non-events are often interpreted in terms of their associated eventive properties. There are two major types for which this is a significant shift in semantic interpretation:

- (2) a Medication as Event:
Orders: Lariam twice daily.
- b Disorder as Event:
Tumor of the left lung.

In both these cases, entities which are not typically marked as events are identified as such, because they contribute significant information to the clinical timeline being constructed. In (2a), for example, the TIMEX3 "twice daily" is interpreted as scoping over the eventuality of the patient *taking* the medication, not the prescription event. In sentence (2b), the "tumor" is interpreted as a stative eventuality of the patient *having* a tumor located within an anatomical region, rather than an entity within an entity.

Within the medical domain, these eventive interpretations of medications, growths and status codes are unambiguous and consistent. Doctors in clinical notes (unlike in biomedical research texts) do

not discuss medications without an associated (implicit) administering EVENT (though some mentions may be hypothetical, generic or negated). Similarly, mentions of symptoms or disorders reflect occurrences in a patient's life, rather than abstract entities. With these interpretations in mind, we can safely infer, for instance, that all UMLS (Unified Medical Language System, (Bodenreider, 2004)) entities of the types Disorder, Chemical/Drug, Procedure and Sign/Symptom will be EVENTS.

In general, in the medical domain, it is essential to read "between the lines" of the shorthand expressions used by the doctors, and recognize implicit events that are being referred to by specific anatomical sites or medications.

4 Modifications to ISO-TimeML for the Clinical Domain

Overall, we have found that the specification required for temporal annotation in the clinical domain does not require substantial modification from existing specifications for the general domain. The clinical domain includes no shortage of inferences, shorthands, and unusual use of language, but the structure of the underlying timeline is not unique.

As a result of this, we have been able to adopt most of the framework from ISO-TimeML, adapting the guidelines where needed, as well as reframing the focus of what gets annotated. This is reflected in a comprehensive guideline, incorporating the specific patterns and uses of events and temporal expressions as seen in clinical data. This approach allows the resulting annotations to be interoperable with existing solutions, while still accommodating the major differences in the nature of the texts. Our guidelines, as well as the annotated data, are available at <http://thyme.healthnlp.org>⁴

Our extensions of the ISO-TimeML specification to the clinical domain are intended to address specific constructions, meanings, and phenomena in medical texts. Our schema differs from ISO-TimeML in a few notable ways.

EVENT Properties We have both simplified the ISO-TimeML coding of EVENTS, and extended it to meet the needs of the clinical domain and the specific language goals of the clinical narrative.

⁴Access to the corpus will require a data use agreement. More information about this process is available from the corpus website.

Consider, for example, how modal subordination is handled in ISO-TimeML. This involves the semantic characterization of an event as “likely”, “possible”, or as presented by observation, evidence, or hearsay. All of these are accounted for compositionally in ISO-TimeML within the SLINK (Subordinating Link) relation (Pustejovsky et al., 2005). While accepting ISO-TimeML’s definition of event modality, we have simplified the annotation task within the current guideline, so that EVENTS now carry attributes for “contextual modality”, “contextual aspect” and “permanence”.

Contextual modality allows the values ACTUAL, HYPOTHETICAL, HEDGED, and GENERIC. ACTUAL covers EVENTS which have actually happened, e.g., “We’ve noted a tumor”. HYPOTHETICAL covers conditionals and possibilities, e.g., “If she develops a tumor”. HEDGED is for situations where doctors proffer a diagnosis, but do so cautiously, to avoid legal liability for an incorrect diagnosis or for overlooking a correct one. For example:

- (3) a. The signal in the MRI is not inconsistent with a **tumor** in the spleen.
- b. The rash appears to be **measles**, awaiting antibody test to confirm.

These HEDGED EVENTS are more real than a hypothetical diagnosis, and likely merit inclusion on a timeline as part of the diagnostic history, but must not be conflated with confirmed fact. These (and other forms of uncertainty in the medical domain) are discussed extensively in (Vincze et al., 2008). In contrast, GENERIC EVENTS do not refer to the patient’s illness or treatment, but instead discuss illness or treatment in general (often in the patient’s specific demographic). For example:

- (4) In other patients without significant **comorbidity** that can **tolerate** adjuvant **chemotherapy**, there is a **benefit** to systemic adjuvant **chemotherapy**.

These sections would be true if pasted into any patient’s note, and are often identical chunks of text repeatedly used to justify a course of action or treatment as well as to defend against liability.

Contextual Aspect (to distinguish from grammatical aspect), allows the clinically-necessary category, INTERMITTENT. This serves to distinguish intermittent EVENTS (such as vomiting or seizures) from constant, more stative EVENTS (such as fever or soreness). For example, the bolded EVENT in (5a) would

be marked as INTERMITTENT, while that in (5b) would not:

- (5) a She has been **vomiting** since June.
- b She has had **swelling** since June.

In the first case, we assume that her vomiting has been intermittent, i.e., there were several points since June in which she was not actively vomiting. In the second case, unless made otherwise explicit (“she has had occasional swelling”), we assume that swelling was a constant state. This property is also used when a particular instance of an EVENT is intermittent, even though it generally would not be:

- (6) Since starting her new regime, she has had occasional bouts of **fever**, but is feeling much better.

The permanence attribute has two values, FINITE and PERMANENT. Permanence is a property of diseases themselves, roughly corresponding to the medical concept of “chronic” vs. “acute” disease, which marks whether a disease is persistent following diagnosis. For example, a (currently) incurable disease like Multiple Sclerosis would be classed as PERMANENT, and thus, once mentioned in a patient’s note, will be assumed to persist through the end of the patient’s timeline. This is compared with FINITE disorders like “Influenza” or “fever”, which, if not mentioned in subsequent notes, should be considered cured and no longer belongs on the patient’s timeline. Because it requires domain-specific knowledge, although present in the specification, Permanence is not currently annotated. However, annotators are trained on the basic idea and told about subsequent axiomatic assignment. The addition of this property to our schema is designed to relieve annotators of any feeling of obligation to express this inferred information in some other way.

TIMEX3 Types Temporal expressions (TIMEX3s) in the clinical domain function the same as in the general linguistic community, with two notable exceptions. ISO-TimeML SETs (statements of frequency) occur quite frequently in the medical domain, particularly with regard to medications and treatments. Medication sections within notes often contain long lists of medications, each with a particular associated set (“Claritin 30mg *twice daily*”), and further temporal specification is not uncommon (e.g., “three times per day at meals”, “once a week at bedtime”).

The second major change for the medical domain is a new type of TIMEX3 which we call PREPOSTEXP. This covers temporally complex terms like

“preoperative”, “postoperative”, and “intraoperative”. These temporal expressions designate a span of time bordered, usually only on one side, by the incorporated event (an operation, in the previous EVENTS). In many cases, the referent is clear:

- (7) She underwent **hemicolectomy** *last week*, and had some postoperative **bleeding**.

Here we understand that “postoperative” refers to “the period of time following the hemicolectomy”. In these cases, the PREPOSTEXP makes explicit a temporal link between the bleeding and the hemicolectomy. In other cases, no clear referent is present:

- (8) Patient shows some **post-procedure** scarring.

In these situations, where no procedure is mentioned (or the reference is never explicitly resolved), we treat the PREPOSTEXP as a narrative container (see Section 5), covering the span of time following the unnamed procedure.

Finally, it is worth noting that the process of normalizing those TIMEX3s is significantly more complex relative to the general domain, because many temporal expressions are anchored not to dates or times, but to other EVENTS (whose dates are often not mentioned or not known by the physician). As we move towards a complete system, we are working to expand the ISO-TimeML system for TIMEX3 normalization to allow some value to be assigned to a phrase like “in the months after her hemicolectomy” when no referent date is present. ISO-TimeML, in discussion with ISO TC 37SC 4, plans to reference to such TIMEX3s in a future release of the standard.

5 Temporal Ordering and Narrative Containers

The semantic content and informational impact of a timeline is encoded in the ordering relations that are identified between the temporal and event expressions present in clinical notes. ISO-TimeML specifies the standard thirteen “Allen relations” from the interval calculus (Allen, 1983), which it refers to as TLINK values. For unguided, general-purpose annotation, the number of relations that could be annotated grows quadratically with the number of events and times, and the task quickly becomes unmanageable. There are, however, strategies that we can adopt to make this labeling task more tractable. Temporal ordering relations in text are of three kinds:

1. Relations between two events
2. Relations between two times

3. Relations between a time and an event.

ISO-TimeML, as a formal specification of the temporal information conveyed in language, makes no distinction between these ordering types. Humans, however, do make distinctions, based on local temporal markers and the discourse relations established in a narrative (Miltakaki et al., 2004; Poesio, 2004).

Because of the difficulty of humans capturing every relationship present in the note (and the disagreement which arises when annotators attempt to do so), it is vital that the annotation guidelines describe an approach that reduces the number of relations that must be considered, but still results in maximally informative temporal links. We have found that many of the weaknesses in prior annotation approaches stem from interaction between two competing goals:

- The guideline should specify certain types of annotations that *should* be performed;
- The guideline should not force annotations to be performed when they need not be.

Failing in the first goal will result in under-annotation and the neglect of relations which provide necessary information for inference and analysis. Failure in the second goal results in over-annotation, creating complex webs of temporal relations which yield mostly inferable information, but which complicate annotation and adjudication considerably.

Our method of addressing both goals in temporal relations annotation is that of the narrative container, discussed in Pustejovsky and Stubbs (2011). A narrative container can be thought of as a temporal bucket into which an EVENT or series of EVENTS may fall, or a natural cluster of EVENTS around a given time or situation. These narrative containers are often represented (or “anchored”) by dates or other temporal expressions (within which a variety of different EVENTS occur), although they can also be anchored to more abstract concepts (“recovery” which might involve a variety of EVENTS) or even durative EVENTS (many other EVENTS can occur during a surgery). Rather than marking every possible TLINK between each EVENT, we instead try to link all EVENTS to their narrative containers, and then link those containers so that the contained EVENTS can be linked by inference.

First, annotators assign each event to one of four broad narrative containers: before the DOCTIME, before and overlapping the DOCTIME, just overlapping the DOCTIME or after the DOCTIME. This narrative

container is identified by the EVENT attribute DocTimeRel. After the assignment of DocTimeRel, the remainder of the narrative container relations must be specified using temporal links (TLINKS). There are five different temporal relations used for such TLINKS: BEFORE, OVERLAP, BEGINS-ON, ENDS-ON and CONTAINS⁵. Due to our narrative container approach, CONTAINS is the most frequent relation by a large margin.

EVENTS serving as narrative container anchors are not tagged as containers per-se. Instead, annotators use the narrative container idea to help them visualize the temporal relations within a document, and then make a series of CONTAINS TLINK annotations which establish EVENTS and TIMEX3s as anchors, and specify their contents. If the annotators do their jobs correctly, properly implementing DocTimeRel and creating accurate TLINKs, a good understanding of the narrative containers present in a document will naturally emerge from the annotated text.

The major advantage introduced with narrative containers is this: a narrative event is placed *within* a bounding temporal interval which is explicitly mentioned in the text. This allows EVENTS within separate containers to be linked by post-hoc inference, temporal reasoning, and domain knowledge, rather than by explicit (and time-consuming) one-by-one temporal relations annotation.

A secondary advantage is that this approach works nicely with the general structure of story-telling in both the general and clinical domains, and provides a compelling and useful metaphor for interpreting timelines. Often, especially in clinical histories, doctors will cluster discussions of symptoms, interventions and diagnoses around a given date (e.g. a whole paragraph starting “June 2009:”), a specific hospitalization (“During her January stay at Mercy”), or a given illness or treatment (“While she underwent Chemo”). Even when specific EVENTS are not explicitly ordered within a cluster (often because the order can be easily inferred with domain knowledge), it is often quite easy to place the EVENTS into containers, and just a few TLINKs can order the containers relative to one another with enough detail to create a clinically useful understanding of the overall timeline.

Narrative containers also allow the inference of relations between sub-events within nested containers:

⁵This is a subset of the ISO-TimeML TLINK types, excluding those seldom occurring in medical records, like ‘simultaneous’ as well as inverse relations like ‘during’ or ‘after’.

(9) December 19th: The patient underwent an **MRI** and **EKG** as well as emergency **surgery**. During the **surgery**, the patient experienced mild **tachycardia**, and she also **bled** significantly during the initial **incision**.

1. December 19th CONTAINS **MRI**
2. December 19th CONTAINS **EKG**
3. December 19th CONTAINS **surgery**
 - a. **surgery** CONTAINS **tachycardia**
 - b. **surgery** CONTAINS **incision**
 - c. **incision** CONTAINS **bled**

Through our container nesting, we can automatically infer that ‘bled’ occurred on December 19th (because ‘19th’ CONTAINS ‘surgery’ which CONTAINS ‘incision’ which CONTAINS ‘bled’). This also allows the capture of EVENT/sub-event relations, and the rapid expression of complex temporal interactions.

6 Explicit vs. Inferable Annotation

Given a specification language, there are essentially two ways of introducing the elements into the document (data source) being annotated:⁶

- Manual annotation: Elements are introduced into the document directly by the human annotator following the guideline.
- Automatic (inferred) annotation: Elements are created by applying an automated procedure that introduces new elements that are derivable from the human annotations.

As such, there is a complex interaction between specification and guideline, and we focus on how the clinical annotation task has helped shape and refine the annotation guidelines. It is important to note that an annotation guideline does not necessarily force the markup of certain elements in a text, even though the specification language (and the eventual goal of the project) might require those annotations to exist.

In some cases, these added annotations are derived logically from human annotations. Explicitly marked temporal relations can be used to infer others that are not marked but exist implicitly through closure. For instance, given EVENTS A, B and C and TLINKS ‘A BEFORE B’ and ‘B BEFORE C’, the TLINK ‘A BEFORE C’ can be automatically inferred. Repeatedly applying such inference rules allows all inferable

⁶We ignore the application of automatic techniques, such as classifiers trained on external datasets, as our focus here is on the preparation of the gold standard used for such classifiers.

TLINKs to be generated (Verhagen, 2005). We can use this idea of closure to show our annotators which annotations need not be marked explicitly, saving time and effort. We have also incorporated these closure rules into our inter-annotator agreement (IAA) calculation for temporal relations, described further in Section 7.2.

The automatic application of rules following the annotation of the text is not limited to the marking of logically inferable relations or EVENTS. In the clinical domain, the combination of within-group shared knowledge and pressure towards concise writing leads to a number of common, inferred relations. Take, for example, the sentence:

(10) Jan 2013: **Colonoscopy, biopsies. Pathology** showed **adenocarcinoma, resected** at Mercy. **Diagnosis T3N1 Adenocarcinoma.**

In this sentence, only the CONTAINS relations between “Jan 2013” and the EVENTS (in bold) are explicitly stated. However, based on the known progression-of-care for colon cancer, we can infer that the colonoscopy occurs first, biopsies occur during the colonoscopy, pathology happens afterwards, a diagnosis (here, adenocarcinoma) is returned after pathology, and resection of the tumor occurs after diagnosis. The presence of the AJCC staging information in the final sentence (along with the confirmation of the adenocarcinoma diagnosis) implies a post-surgical pathology exam of the resected specimen, as the AJCC staging information cannot be determined without this additional examination.

These inferences come naturally to domain experts but are largely inaccessible to people outside the medical community without considerable annotator training. Making explicit our understanding of these “understood orderings” is crucial; although they are not marked by human annotators in our schema, the annotators often found it initially frustrating to leave these (purely inferential) relations unstated. Although many of our (primarily linguistically trained) annotators learned to see these patterns, we chose to exclude them from the manual task since newer annotators with varying degrees of domain knowledge may struggle if asked to manually annotate them.

Similar unspoken-but-understood orderings are found throughout the clinical domain. As mentioned in Section 3, both Permanence and Contextual Aspect:Intermittent are properties of symptoms and diseases themselves, rather than of the patient’s particular situation. As such, these properties could easily

Annotation Type	Raw Count
EVENT	15,769
TIMEX3	1,426
LINK	7935
Total	25,130

Table 1: Raw Frequency of Annotation Types

TLINK Type	Raw Count	% of TLINKS
CONTAINS	5,112	64.42%
OVERLAP	1,205	15.19%
BEFORE	1,004	12.65%
BEGINS-ON	488	6.15%
ENDS-ON	126	1.59%
Total	7,935	100.00%

Table 2: Relative Frequency of TLINK types

be identified and marked across a medical ontology, and then be automatically assigned to EVENTS recognized as specific medical named entities.

Finally, due to the peculiarities of EHR systems, some annotations *must* be done programmatically. Exact dates of patient visit (or of pathology/radiology consult) are often recorded as metadata on the EHR itself, rather than within the text, making the canonical DOCTIME (or time of automatic section modifications) difficult to access in de-identified plaintext data, but easy to find automatically.

7 Results

We report results on the annotations from the here-released subset of the THYME colon cancer corpus, which includes clinical notes and pathology reports for 35 patients diagnosed with colon cancer for a total of 107 documents. Each note was annotated by a pair of graduate or undergraduate students in Linguistics at the University of Colorado, then adjudicated by a domain expert. These clinical narratives were sampled from the EHRs of a major healthcare center (the Mayo Clinic). They were deidentified for all patient-sensitive information; however, original dates were retained.

7.1 Descriptive Statistics

Table 1 presents the raw counts for events, temporal expressions and links in the adjudicated gold annotations. Table 2 presents the number and percentage of TLINKs by type in the adjudicated relations gold annotations.

Annotation Type	F1-Score	Alpha
EVENT	0.8038	0.7899
TIMEX3	0.8047	0.6705
LINK: Participants only	0.5012	0.4999
LINK: Participants+type	0.4506	0.4503
LINK: CONTAINS	0.5630	0.5626

Table 3: IAA (F1-Score and Alpha) by annotation type

EVENT Property	F1-Score	Alpha
DocTimeRel	0.7189	0.6889
Cont.Aspect	0.9947	0.9930
Cont.Modality	0.9547	0.9420

Table 4: IAA (F1-Score and Alpha) for EVENT properties

7.2 Inter-annotator Agreement

We report inter-annotator agreement (IAA) results on the THYME corpus. Each note was annotated by two independent annotators. The final gold standard was produced after disagreement adjudication by a third annotator was performed.

We computed the IAA as F1-score and Krippendorff’s Alpha (Krippendorff, 2012) by applying closure, using explicitly marked temporal relations to identify others that are not marked but exist implicitly. In the computation of the IAA, inferred-only TLINKs do not contribute to the score, matched or unmatched. For instance, if both annotators mark A BEFORE B and B BEFORE C, to prevent artificially inflating the agreement score, the inferred A BEFORE C is ignored. Likewise, if one annotator marked A BEFORE B and B BEFORE C and the other annotator did not, the inferred A BEFORE C is not counted. However, if one annotator did explicitly mark A BEFORE C, then an equivalent inferred TLINK would be used to match it. EVENT and TIMEX3 IAA was generated based on exact and overlapping spans, respectively. These results are reported in Table 3.

The THYME corpus also differs from ISO-TimeML in terms of EVENT properties, with the addition of DocTimeRel, ContextualModality and ContextualAspect. IAA for these properties is in Table 4.

7.3 Baseline Systems

To get an idea of how much work will be necessary to adapt existing temporal information extraction systems to the clinical domain, we took the freely available ClearTK-TimeML system (Bethard, 2013),

	TempEval 2013			THYME Corpus		
	<i>P</i>	<i>R</i>	<i>F</i> ₁	<i>P</i>	<i>R</i>	<i>F</i> ₁
TIMEX3	83.2	71.7	77.0	59.3	42.8	49.7
EVENT	81.4	76.4	78.8	78.9	23.9	36.6
DocTimeRel	-	-	-	47.4	47.4	47.4
LINK ⁷	28.6	30.9	26.6	22.7	18.6	20.4
EVENT-TIMEX3	-	-	-	32.3	60.7	42.1
EVENT-EVENT	-	-	-	7.0	3.0	4.2

Table 5: Performance of ClearTK-TimeML models, as reported in the TempEval 2013 competition, and as applied to the THYME Corpus development set.

which was among the top performing systems in TempEval 2013 (UzZaman et al., 2013), and evaluated its performance on the THYME corpus.

ClearTK-TimeML uses support vector machine classifiers trained on the TempEval 2013 training data, employing a small set of features including character patterns, tokens, stems, part-of-speech tags, nearby nodes in the constituency tree, and a small time word gazetteer. For EVENTS and TIMEX3s, the ClearTK-TimeML system could be applied directly to the THYME corpus. For DocTimeRels, the relation for an EVENT was taken from the TLINK between that EVENT and the document creation time, after mapping INCLUDES to OVERLAP. EVENTS with no such TLINK were assumed to have a DocTimeRel of OVERLAP. For other temporal relations, INCLUDES was mapped to CONTAINS.

Results of this system on TempEval 2013 and the THYME corpus are shown in Table 5. For time expressions, performance when moving to the clinical data degrades about 25%, from *F*₁ of 77.0 to 49.7. For events, the degradation is much larger, about 40%, from 78.8 to 36.6, most likely because of the large number of clinical symptoms, diseases, disorders, etc. which have never been observed by the system during training. Temporal relations are a bit more difficult to compare because TempEval lumped DocTimeRel and other temporal relations together and had several differences in their evaluation metric⁷. However, we at least can see that performance of the ClearTK-TimeML system on temporal relations is low on clinical text, achieving only *F*₁ of 20.4.

These results suggest that clinical narratives do

⁷The TempEval 2013 evaluation metric penalized systems for parts of the text that were not examined by annotators, and used different variants of closure-based precision and recall.

indeed present new challenges for temporal information extraction systems, and that having access to domain specific training data will be crucial for accurate extraction in the clinical domain. At the same time, it is encouraging that we were able to apply existing ISO-TimeML-based systems to our corpus, despite the several extensions to ISO-TimeML that were necessary for clinical narratives.

8 Discussion

CONTAINS plays a large role in the THYME corpus, representing 66% of TLINK annotations made, compared with only 14.6% for OVERLAP, the second most frequent type. We also see that BEFORE links are relatively less common than OVERLAP and CONTAINS, illustrating that much of the temporal ordering on the timeline is accomplished by using many vertical links (CONTAINS, OVERLAP) to build containers, and few horizontal links (BEFORE, BEGINS-ON, ENDS-ON) to order them.

IAA on EVENTS and Temporal Expressions is strong, although differentiating implicit EVENTS (which should not be marked) from explicit, markable EVENTS remains one of the biggest sources of disagreement. When compared to the data from the 2012 i2b2 challenge (Sun et al., 2013b), our IAA figures are quite similar. Even with our more complex schema, we achieved an F1-score of 0.8038 for EVENTS (compared to the i2b2 score of 0.87 for partial match). For TIMEX3s, our F1-score was 0.8047, compared to an F1-score of 0.89 for i2b2.

TLINKing medical EVENTS remains a very difficult task. By using our narrative container approach to constrain the number of necessary annotations and by eliminating often-confusing inverse relations (like ‘after’ and ‘during’) (neither of which were done for the i2b2 data), we were able to significantly improve on the i2b2 TLINK span agreement F1-score of 0.39, achieving an agreement score of 0.5012 for all LINKS across our corpus. The majority of remaining annotator disagreement comes from different opinions about whether any two EVENTS require an explicit TLINK between them or an inferred one, rather than what type of TLINK it would be (e.g. BEFORE vs. CONTAINS). Although our results are still significantly higher than the results reported for i2b2, and in line with previously reported general news figures, we are not satisfied. Improving IAA is an important goal for future work, and with further training, specification, experience, and standardization, we hope to

clarify contexts for explicit TLINKS.

News-trained temporal information extraction systems see a significant drop in performance when applied to the clinical texts of the THYME corpus. But as the corpus is an extension of ISO-TimeML, future work will be able to train ISO-TimeML compliant systems on the annotations of the THYME corpus to reduce or eliminate this performance gap.

Some applications that our work may enable include (1) better understanding of event semantics, such as whether a disease is chronic or acute and its usual natural history, (2) typical event duration for these events, (3) the interaction of general and domain-specific events and their importance in the final timeline, and, more generally, (4) the importance of rough temporality and narrative containers as a step towards finer-grained timelines.

We have several avenues of ongoing and future work. First, we are working to demonstrate the utility of the THYME corpus for training machine learning models. We have designed support vector machine models with constituency tree kernels that were able to reach an F1-score of 0.737 on an EVENT-TIMEX3 narrative container identification task (Miller et al., 2013), and we are working on training models to identify events, times and the remaining types of temporal relations. Second, as per our motivating use cases, we are working to integrate this annotation data with timeline visualization tools and to use these annotations in quality-of-care research. For example, we are using temporal reasoning built on this work to investigate the liver toxicity of methotrexate across a large corpus of EHRs (Lin et al., under review)]. Finally, we plan to explore the application of our notion of an event (anything that should be visible on a domain-appropriate timeline) to other domains. It should transfer naturally to clinical notes about other (non-cancer) conditions, and even to other types of clinical notes, as certain basic events should always be included in a patient’s timeline. Applying our notion of event to more distant domains, such as legal opinions, would require first identifying a consensus within the domain about which events must appear on a timeline.

9 Conclusion

Much of the information in clinical notes critical to the construction of a detailed timeline is left implicit by the concise shorthand used by doctors. Many events are referred to only by a term such as “tu-

mor”, while properties of the event itself, such as “intermittent”, may not be specified. In addition, the ordering of events on a timeline is often left to the reader to infer, based on domain-specific knowledge. It is incumbent upon the annotation guideline to indicate that only informative event orderings should be annotated, while leaving domain-specific orderings to post-annotation inference. This document has detailed our approach to adapting the existing ISO-TimeML standard to this recovery of implicit information, and defining guidelines that support annotation within this complex domain. Our guidelines, as well as the annotated data, are available at <http://thyme.healthnlp.org>, and the full corpus has been proposed for use in a SemEval 2015 shared task.

Acknowledgments

The project described is supported by Grant Number R01LM010090 and U54LM008748 from the National Library Of Medicine. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Library Of Medicine or the National Institutes of Health.

We would also like to thank Dr. Piet C. de Groen and Dr. Brad Erickson at the Mayo Clinic, as well as Dr. William F. Styler III, for their contributions to the schema and to our understanding of the intricacies of clinical language.

References

- James F Allen. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.
- Emmon Bach. 1986. The algebra of events. *Linguistics and philosophy*, 9(1):5–16.
- Steven Bethard. 2013. Cleartk-timeml: A minimalist approach to tempeval 2013. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 10–14, Atlanta, Georgia, USA, June. Association for Computational Linguistics.
- Olivier Bodenreider. 2004. The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic acids research*, 32(Database issue):D267–D270, January.
- Philip Bramsen, Pawan Deshpande, Yoong Keok Lee, and Regina Barzilay. 2006. Finding temporal order in discharge summaries. In *AMIA Annual Symposium Proceedings*, volume 2006, page 81. American Medical Informatics Association.
- Carlo Combi, Yuval Shahar, et al. 1997. Temporal reasoning and temporal data maintenance in medicine: issues and challenges. *Computers in biology and medicine*, 27(5):353–368.
- Robert H Dolin. 1995. Modeling the temporal complexities of symptoms. *Journal of the American Medical Informatics Association*, 2(5):323–331.
- George Hripesak, Nicholas D Soulakakis, Li Li, Frances P Morrison, Albert M Lai, Carol Friedman, Neil S Calman, and Farzad Mostashari. 2009. Syndromic surveillance using ambulatory electronic health records. *Journal of the American Medical Informatics Association*, 16(3):354–361.
- Ann K Irvine, Stephanie W Haas, and Tessa Sullivan. 2008. Tn-ties: A system for extracting temporal information from emergency department triage notes. In *AMIA Annual Symposium proceedings*, volume 2008, page 328. American Medical Informatics Association.
- Elpida T Keravnou. 1997. Temporal abstraction of medical data: Deriving periodicity. In *Intelligent Data Analysis in Medicine and Pharmacology*, pages 61–79. Springer.
- Klaus H. Krippendorff. 2012. *Content Analysis: An Introduction to Its Methodology*. SAGE Publications, Inc, third edition edition, April.
- Chen Lin, Elizabeth Karlson, Dmitriy Dligach, Monica Ramirez, Timothy Miller, Huan Mo, Natalie Braggs, Andrew Cagan, Joshua Denny, and Guergana Savova. under review. Automatic identification of methotrexade-induced liver toxicity in rheumatoid arthritis patients from the electronic medical records. *Journal of the Medical Informatics Association*.
- John McCarthy. 2002. Actions and other events in situation calculus. In *Proceedings of the International conference on Principles of Knowledge Representation and Reasoning*, pages 615–628. Morgan Kaufmann Publishers; 1998.
- Stéphane M Meystre, Guergana K Savova, Karin C Kipper-Schuler, John F Hurdle, et al. 2008. Extracting information from textual documents in the electronic health record: a review of recent research. *Yearb Med Inform*, 35:128–44.
- Timothy Miller, Steven Bethard, Dmitriy Dligach, Sameer Pradhan, Chen Lin, and Guergana Savova. 2013. Discovering temporal narrative containers in clinical text. In *Proceedings of the 2013 Workshop on Biomedical Natural Language Processing*, pages 18–26, Sofia, Bulgaria, August. Association for Computational Linguistics.

- Eleni Miltsakaki, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2004. The penn discourse treebank. In *In Proceedings of LREC 2004*.
- Massimo Poesio. 2004. Discourse annotation and semantic annotation in the gnome corpus. In *In Proceedings of the ACL Workshop on Discourse Annotation*.
- James Pustejovsky and Amber Stubbs. 2011. Increasing informativeness in temporal annotation. In *Proceedings of the 5th Linguistic Annotation Workshop*, pages 152–160. Association for Computational Linguistics.
- James Pustejovsky, Robert Knippen, Jessica Littman, and Roser Sauri. 2005. Temporal and event information in natural language text. *Language Resources and Evaluation*, 39(2-3):123–164.
- James Pustejovsky, Kiyong Lee, Harry Bunt, and Laurent Romary. 2010. Iso-timeml: An international standard for semantic annotation. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC 2010)*, Valletta, Malta.
- Angus Roberts, Robert Gaizauskas, Mark Hepple, George Demetriou, Yikun Guo, and Ian Roberts. 2009. Building a semantically annotated corpus of clinical texts. *Journal of biomedical informatics*, 42(5):950–966.
- Guergana Savova, Steven Bethard, Will Styler, James Martin, Martha Palmer, James Masanz, and Wayne Ward. 2009. Towards temporal relation discovery from the clinical narrative. In *AMIA Annual Symposium Proceedings*, volume 2009, page 568. American Medical Informatics Association.
- Tessa Sullivan, Ann Irvine, and Stephanie W Haas. 2008. It’s all relative: usage of relative temporal expressions in triage notes. *Proceedings of the American Society for Information Science and Technology*, 45(1):1–8.
- Weiyi Sun, Anna Rumshisky, and Ozlem Uzuner. 2013a. Evaluating temporal relations in clinical text: 2012 i2b2 challenge. *Journal of the American Medical Informatics Association*.
- Weiyi Sun, Anna Rumshisky, and Ozlem Uzuner. 2013b. Evaluating temporal relations in clinical text: 2012 i2b2 challenge. *Journal of the American Medical Informatics Association*, 20(5):806–813.
- Alexander Turchin, Maria Shubina, Eugene Breydo, Merri L Pendergrass, and Jonathan S Einbinder. 2009. Comparison of information content of structured and narrative text data sources on the example of medication intensification. *Journal of the American Medical Informatics Association*, 16(3):362–370.
- Naushad UzZaman, Hector Llorens, Leon Derczynski, James Allen, Marc Verhagen, and James Pustejovsky. 2013. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 1–9, Atlanta, Georgia, USA, June. Association for Computational Linguistics.
- Marc Verhagen. 2005. Temporal Closure in an Annotation Environment. *Language Resources and Evaluation*, 39(2):211–241.
- Veronika Vincze, Gyrgy Szarvas, Richrd Farkas, Gyrgy Mra, and Jnos Csirik. 2008. The bioscope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC Bioinformatics*, 9(Suppl 11):1–9.
- Ying Zhao, George Karypis, and Usama M. Fayyad. 2005. Hierarchical clustering algorithms for document datasets. *Data Mining and Knowledge Discovery*, 10:141–168.
- Jiaping Zheng, Wendy W Chapman, Rebecca S Crowley, and Guergana K Savova. 2011. Coreference resolution: A review of general methodologies and applications in the clinical domain. *Journal of biomedical informatics*, 44(6):1113–1122.
- Li Zhou, Simon Parsons, and George Hripcsak. 2008. The evaluation of a temporal reasoning system in processing clinical discharge summaries. *Journal of the American Medical Informatics Association*, 15(1):99–106.

Entity Linking meets Word Sense Disambiguation: a Unified Approach

Andrea Moro, Alessandro Raganato, Roberto Navigli

Dipartimento di Informatica,
Sapienza Università di Roma,
Viale Regina Elena 295, 00161 Roma, Italy
{moro,navigli}@di.uniroma1.it
ale.raganato@gmail.com

Abstract

Entity Linking (EL) and Word Sense Disambiguation (WSD) both address the lexical ambiguity of language. But while the two tasks are pretty similar, they differ in a fundamental respect: in EL the textual mention can be linked to a named entity which may or may not contain the exact mention, while in WSD there is a perfect match between the word form (better, its lemma) and a suitable word sense.

In this paper we present Babelfy, a unified graph-based approach to EL and WSD based on a loose identification of candidate meanings coupled with a densest subgraph heuristic which selects high-coherence semantic interpretations. Our experiments show state-of-the-art performances on both tasks on 6 different datasets, including a multilingual setting. Babelfy is online at <http://babelfy.org>

1 Introduction

The automatic understanding of the meaning of text has been a major goal of research in computational linguistics and related areas for several decades, with ambitious challenges, such as Machine Reading (Etzioni et al., 2006) and the quest for knowledge (Schubert, 2006). Word Sense Disambiguation (WSD) (Navigli, 2009; Navigli, 2012) is a historical task aimed at assigning meanings to single-word and multi-word occurrences within text, a task which is more alive than ever in the research community.

Recently, the collaborative creation of large semi-structured resources, such as Wikipedia, and knowledge resources built from them (Hovy et al., 2013),

such as BabelNet (Navigli and Ponzetto, 2012a), DBpedia (Auer et al., 2007) and YAGO2 (Hoffart et al., 2013), has favoured the emergence of new tasks, such as Entity Linking (EL) (Rao et al., 2013), and opened up new possibilities for tasks such as Named Entity Disambiguation (NED) and Wikification. The aim of EL is to discover mentions of entities within a text and to link them to the most suitable entry in a reference knowledge base. However, in contrast to WSD, a mention may be partial while still being unambiguous thanks to the context. For instance, consider the following sentence:

(1) Thomas and Mario are strikers playing in Munich.

This example makes it clear how intertwined the two tasks of WSD and EL are. In fact, on the one hand, *striker* and *play* are polysemous words which can be disambiguated by selecting the game/soccer playing senses of the two words in a dictionary; on the other hand, *Thomas* and *Mario* are partial mentions which have to be linked to the appropriate entries of a knowledge base, that is, *Thomas Müller* and *Mario Gomez*, two well-known soccer players.

The two main differences between WSD and EL lie, on the one hand, in the kind of inventory used, i.e., dictionary vs. encyclopedia, and, on the other hand, in the assumption that the mention is complete or potentially partial. Notwithstanding these differences, the tasks are similar in nature, in that they both involve the disambiguation of textual fragments according to a reference inventory. However, the research community has so far tackled the two tasks separately, often duplicating efforts and solutions.

In contrast to this trend, research in knowledge acquisition is now heading towards the seamless in-

tegration of encyclopedic and lexicographic knowledge into structured language resources (Hovy et al., 2013), and the main representative of this new direction is undoubtedly BabelNet (Navigli and Ponzetto, 2012a). Given such structured language resources it seems natural to suppose that they might provide a common ground for the two tasks of WSD and EL.

More precisely, in this paper we explore the hypothesis that the lexicographic knowledge used in WSD is also useful for tackling the EL task, and vice versa, that the encyclopedic information utilized in EL helps disambiguate nominal mentions in a WSD setting. We propose Babelfy, a novel, unified graph-based approach to WSD and EL, which performs two main steps: i) it exploits random walks with restart, and triangles as a support for reweighting the edges of a large semantic network; ii) it uses a densest subgraph heuristic on the available semantic interpretations of the input text to perform a joint disambiguation with both concepts and named entities. Our experiments show the benefits of our synergistic approach on six gold-standard datasets.

2 Related Work

2.1 Word Sense Disambiguation

Word Sense Disambiguation (WSD) is the task of choosing the right sense for a word within a given context. Typical approaches for this task can be classified as i) supervised, ii) knowledge-based, and iii) unsupervised. However, supervised approaches require huge amounts of annotated data (Zhong and Ng, 2010; Shen et al., 2013; Pilehvar and Navigli, 2014), an effort which cannot easily be repeated for new domains and languages, while unsupervised ones suffer from data sparsity and an intrinsic difficulty in their evaluation (Agirre et al., 2006; Brody and Lapata, 2009; Manandhar et al., 2010; Van de Cruys and Apidianaki, 2011; Di Marco and Navigli, 2013). On the other hand, knowledge-based approaches are able to obtain good performance using readily-available structured knowledge (Agirre et al., 2010; Guo and Diab, 2010; Ponzetto and Navigli, 2010; Miller et al., 2012; Agirre et al., 2014). Some of these approaches marginally take into account the structural properties of the knowledge base (Mihalcea, 2005). Other approaches, instead, leverage the structural properties of the knowledge base

by exploiting centrality and connectivity measures (Sinha and Mihalcea, 2007; Tsatsaronis et al., 2007; Agirre and Soroa, 2009; Navigli and Lapata, 2010).

One of the key steps of many knowledge-based WSD algorithms is the creation of a graph representing the semantic interpretations of the input text. Two main strategies to build this graph have been proposed: i) exploiting the direct connections, i.e., edges, between the considered sense candidates; ii) populating the graph according to (shortest) paths between them. In our approach we manage to unify these two strategies by automatically creating edges between sense candidates performing Random Walk with Restart (Tong et al., 2006).

The recent upsurge of interest in multilinguality has led to the development of cross-lingual and multilingual approaches to WSD (Lefever and Hoste, 2010; Lefever and Hoste, 2013; Navigli et al., 2013). Multilinguality has been exploited in different ways, e.g., by using parallel corpora to build multilingual contexts (Guo and Diab, 2010; Banea and Mihalcea, 2011; Lefever et al., 2011) or by means of ensemble methods which exploit complementary sense evidence from translations in different languages (Navigli and Ponzetto, 2012b). In this work, we present a novel exploitation of the structural properties of a multilingual semantic network.

2.2 Entity Linking

Entity Linking (Erbs et al., 2011; Rao et al., 2013; Cornolti et al., 2013) encompasses a set of similar tasks, which include Named Entity Disambiguation (NED), that is the task of linking entity mentions in a text to a knowledge base (Bunescu and Pasca, 2006; Cucerzan, 2007), and Wikification, i.e., the automatic annotation of text by linking its relevant fragments of text to the appropriate Wikipedia articles. Mihalcea and Csomai (2007) were the first to tackle the Wikification task. In their approach they disambiguate each word in a sentence independently by exploiting the context in which it occurs. However, this approach is local in that it lacks a collective notion of coherence between the selected Wikipedia pages. To overcome this problem, Cucerzan (2007) introduced a global approach based on the simultaneous disambiguation of all the terms in a text and the use of lexical context to disambiguate the mentions. To maximize the semantic agreement Milne

and Witten (2008) introduced the analysis of the semantic relations between the candidate senses and the unambiguous context, i.e., words with a single sense candidate. However, the performance of this algorithm depends heavily on the number of links incident to the target senses and on the availability of unambiguous words within the input text. To overcome this issue a novel class of approaches have been proposed (Kulkarni et al., 2009; Ratinov et al., 2011; Hoffart et al., 2011) that exploit global and local features. However, these systems either rely on a difficult NP-hard formalization of the problem which is infeasible for long text, or exploit popularity measures which are domain-dependent. In contrast, we show that the semantic network structure can be leveraged to obtain state-of-the-art performance by synergistically disambiguating both word senses and named entities at the same time.

Recently, the explosion of on-line social networking services, such as Twitter and Facebook, have contributed to the development of new methods for the efficient disambiguation of short texts (Ferragina and Scaiella, 2010; Hoffart et al., 2012; Böhm et al., 2012). Thanks to a loose candidate identification technique coupled with a densest subgraph heuristic, we show that our approach is particularly suited for short and highly ambiguous text disambiguation.

2.3 The Best of Two Worlds

Our main goal is to bring together the two worlds of WSD and EL. On the one hand, this implies relaxing the constraint of a perfect association between mentions and meanings, which is, instead, assumed in WSD. On the other hand, this relaxation leads to the inherent difficulty of encoding a full-fledged sense inventory for EL. Our solution to this problem is to keep the set of candidate meanings for a given mention as open as possible (see Section 6), so as to enable high recall in linking partial mentions, while providing an effective method for handling this high ambiguity (see Section 7).

A key assumption of our work is that the lexicographic knowledge used in WSD is also useful for tackling the EL task, and vice versa the encyclopedic information utilized in EL helps disambiguate nominal mentions in a WSD setting. We enable the joint treatment of concepts and named entities by enforcing high coherence in our semantic interpretations.

3 WSD and Entity Linking Together

Task. Our task is to disambiguate and link all nominal and named entity mentions occurring within a text. The linking task is performed by associating each mention with the most suitable entry of a given knowledge base.¹

We point out that our definition is unconstrained in terms of what to link, i.e., unlike Wikification and WSD, we can link overlapping fragments of text. For instance, given the text fragment *Major League Soccer*, we identify and disambiguate several different nominal and entity mentions: *Major League Soccer*, *major league*, *league* and *soccer*. In contrast to EL, we link not only named entity mentions, such as *Major League Soccer*, but also nominal mentions, e.g., *major league*, to their corresponding meanings in the knowledge base.

Babelfy. We provide a unified approach to WSD and entity linking in three steps:

1. Given a lexicalized semantic network, we associate with each vertex, i.e., either concept or named entity, a semantic signature, that is, a set of related vertices (Section 5). This is a preliminary step which needs to be performed only once, independently of the input text.
2. Given a text, we extract all the linkable fragments from this text and, for each of them, list the possible meanings according to the semantic network (Section 6).
3. We create a graph-based semantic interpretation of the whole text by linking the candidate meanings of the extracted fragments using the previously-computed semantic signatures. We then extract a dense subgraph of this representation and select the best candidate meaning for each fragment (Section 7).

4 Semantic Network

Our approach requires the availability of a wide-coverage semantic network which encodes structural and lexical information both of an encyclopedic and of a lexicographic kind. Although in principle any semantic network with these properties

¹Mentions which are not contained in the reference knowledge base are not taken into account.

could be utilized, in our work we used the BabelNet² 1.1.1 semantic network (Navigli and Ponzetto, 2012a) since it is the largest multilingual knowledge base, obtained from the automatic seamless integration of Wikipedia³ and WordNet (Fellbaum, 1998). We consider BabelNet as a directed multigraph which contains both concepts and named entities as its vertices and a multiset of semantic relations as its edges. We leverage the multilingual lexicalizations of the vertices of BabelNet to identify mentions in the input text. For example, the entity *FC Bayern Munich* can be lexicalized in different languages, e.g., *F.C. Bayern de Múnich* in Spanish, *Die Roten* in English and *Bayern München* in German, among others. As regards semantic relations, the only information we use is that of the end points, i.e., vertices, that these relations connect, while neglecting the relation type.

5 Building Semantic Signatures

One of the major issues affecting both manually-curated and automatically constructed semantic networks is data sparsity. For instance, we calculated that the average number of incident edges is roughly 10 in WordNet, 50 in BabelNet and 80 in YAGO2, to mention a few. Although automatically-built resources typically provide larger amounts of edges, two issues have to be taken into account: concepts which should be related might not be directly connected despite being structurally close within the network, and, vice versa, weakly-related or even unrelated concepts can be erroneously connected by an edge. For instance, in BabelNet we do not have an edge between *playmaker* and *Thomas Müller*, while we have an incorrect edge connecting *FC Bayern Munich* and *Yellow Submarine (song)*. However, this crisp notion of relatedness can be overcome by exploiting the global structure of the semantic network, thereby obtaining a more precise and higher-coverage measure of relatedness. We address this issue in two steps: first, we provide a structural weighting of the network’s edges; second, for each vertex we create a set of related vertices using random walks with restart.

²<http://babelnet.org>

³<http://www.wikipedia.org>

Structural weighting. Our first objective is to assign higher weights to edges which are involved in more densely connected areas of the directed network. To this end, inspired by the local clustering coefficient measure (Watts and Strogatz, 1998) and its recent success in Word Sense Induction (Di Marco and Navigli, 2013), we use directed triangles, i.e., directed cycles of length 3, and weight each edge (v, v') by the number of directed triangles it occurs in:

$$weight(v, v') := |\{(v, v', v'') : (v, v'), (v', v''), (v'', v) \in E\}| + 1 \quad (1)$$

We add one to each weight to ensure the highest degree of reachability in the network.

Random Walk with Restart. Our goal is to create a *semantic signature* (i.e., a set of highly related vertices) for each concept and named entity of the semantic network. To do this, we perform a Random Walk with Restart (RWR) (Tong et al., 2006), that is, a stochastic process that starts from an initial vertex of the graph⁴ and then, for a fixed number n of steps or until convergence, explores the graph by choosing the next vertex within the current neighborhood or by restarting from the initial vertex with a given, fixed *restart probability* α . For each edge (v, v') in the network, we model the conditional probability $P(v'|v)$ as the normalized weight of the edge:

$$P(v'|v) = \frac{weight(v, v')}{\sum_{v'' \in V} weight(v, v'')}$$

where V is the set of vertices of the semantic network and $weight(v, v')$ is the function defined in Equation 1. We then run the RWR from each vertex v of the semantic network for a fixed number n of steps (we show in Algorithm 1 our RWR pseudocode). We keep track of the encountered vertices using the map *counts*, i.e., we increase the counter associated with vertex v' in *counts* every time we hit v' during a RWR started from v (see line 11). As a result, we obtain a frequency distribution over the whole set of concepts and entities. To eliminate weakly-related vertices we keep only those items that were hit at least η times (see lines 16–18). Finally, we save the remaining vertices in the set *semSign_v* which is the semantic signature of v (see line 19).

⁴RWR can be used with an initial set of vertices, however in this paper we use a single initial vertex.

Algorithm 1 Random walk with restart.

```
1: input:  $v$ , the starting vertex;  
    $\alpha$ , the restart probability;  
    $n$ , the number of steps to be executed;  
    $P$ , the transition probabilities;  
    $\eta$ , the frequency threshold.  
2: output:  $semSign_v$ , set of related vertices for  $v$ .  
3: function RWR( $v, \alpha, n, P, \eta$ )  
4:    $v' := v$   
5:    $counts := \text{new Map} < Synset, Integer >$   
6:   while  $n > 0$  do  
7:     if  $random() > \alpha$  then  
8:       given the transition probabilities  $P(\cdot|v')$   
9:       of  $v'$ , choose a random neighbor  $v''$   
10:       $v' := v''$   
11:       $counts[v'] ++$   
12:     else  
13:       restart the walk  
14:       $v' := v$   
15:       $n := n - 1$   
16:   for each  $v'$  in  $counts.keys()$  do  
17:     if  $counts[v'] < \eta$  then  
18:       remove  $v'$  from  $counts.keys()$   
19:   return  $semSign_v = counts.keys()$ 
```

The creation of our set of semantic signatures, one for each vertex in the semantic network, is a preliminary step carried out once only before starting processing any input text. We now turn to the candidate identification and disambiguation steps.

6 Candidate Identification

Given a text as input, we apply part-of-speech tagging and identify the set F of all the textual fragments, i.e., all the sequences of words of maximum length five, which contain at least one noun and that are substrings of lexicalizations in BabelNet, i.e., those fragments that can potentially be linked to an entry in BabelNet. For each textual fragment $f \in F$, i.e., a single- or multi-word expression of the input text, we look up the semantic network for candidate meanings, i.e., vertices that contain f or, only for named entities, a superstring of f as their lexicalization. For instance, for sentence (1) in the introduction, we identify the following textual fragments: *Thomas, Mario, strikers, Munich*. This output is obtained thanks to our loose candidate identification routine, i.e., based on superstring matching instead of exact matching, which, for instance, enables us to recognize the right candidate *Mario Gomez* for the

mention *Mario* even if this named entity does not have *Mario* as one of its lexicalizations (for an analysis of the impact of this routine against the exact matching approach see the discussion in Section 9).

Moreover, as we stated in Section 3, we allow overlapping fragments, e.g., for *major league* we recognize *league* and *major league*. We denote with $cand(f)$ the set of all the candidate meanings of fragment f . For instance, for the noun *league* we have that $cand(league)$ contains among others the *sport* word sense and the *TV series* named entity.

7 Candidate Disambiguation

Semantic interpretation graph. After the identification of fragments (F) and their candidate meanings ($cand(\cdot)$), we create a directed graph $G_I = (V_I, E_I)$ of the semantic interpretations of the input text. We show the pseudocode in Algorithm 2. V_I contains all the candidate meanings of all fragments, that is, $V_I := \{(v, f) : v \in cand(f), f \in F\}$, where f is a fragment of the input text and v is a candidate Babel synset that has a lexicalization which is equal to or is a superstring of f (see lines 4–8). The set of edges E_I connects related meanings and is populated as follows: we add an edge from (v, f) to (v', f') if and only if $f \neq f'$ and $v' \in semSign_v$ (see lines 9–11). In other words, we connect two candidate meanings of different fragments if one is in the semantic signature of the other. For instance, we add an edge between $(Mario\ Gomez, Mario)$ and $(Thomas\ M\ddot{u}ller, Thomas)$, while we do not add one between $(Mario\ Gomez, Mario)$ and $(Mario\ Basler, Mario)$ since these are two candidate meanings of the same fragment, i.e., *Mario*. In Figure 1, we show an excerpt of our graph for sentence (1).

At this point we have a graph-based representation of all the possible interpretations of the input text. In order to drastically reduce the degree of ambiguity while keeping the interpretation coherence as high as possible, we apply a novel densest subgraph heuristic (see line 12), whose description we defer to the next paragraph. The result is a subgraph which contains those semantic interpretations that are most coherent to each other. However, this subgraph might still contain multiple interpretations for the same fragment, and even unambiguous fragments which are not correct. Therefore, the final

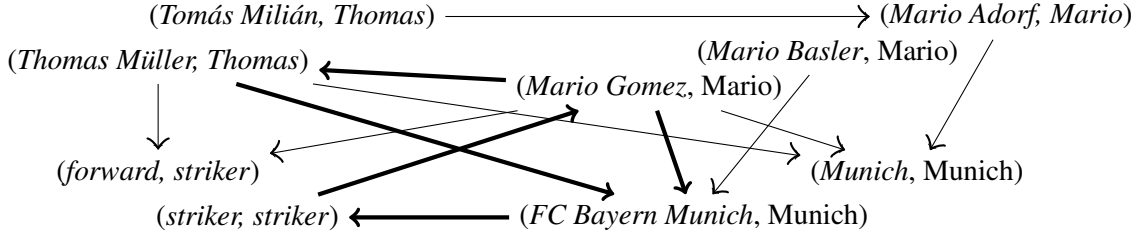


Figure 1: An excerpt of the semantic interpretation graph automatically built for the sentence *Thomas and Mario are strikers playing in Munich* (the edges connecting the correct meanings are in bold).

step is the selection of the most suitable candidate meaning for each fragment f given a threshold θ to discard semantically unrelated candidate meanings. We score each meaning $v \in \text{cand}(f)$ with its normalized weighted degree⁵ in the densest subgraph:

$$\text{score}((v, f)) = \frac{w_{(v, f)} \cdot \text{deg}((v, f))}{\sum_{v' \in \text{cand}(f)} w_{(v', f)} \cdot \text{deg}((v', f))} \quad (2)$$

where $w_{(v, f)}$ is the fraction of fragments the candidate meaning v connects to:

$$w_{(v, f)} := \frac{|\{f' \in F : \exists v' \text{ s.t. } ((v, f), (v', f')) \text{ or } ((v', f'), (v, f)) \in E_I\}|}{|F| - 1}$$

The rationale behind this scoring function is to take into account both the semantic coherence, using a graph centrality measure among the candidate meanings, and the lexical coherence, in terms of the number of fragments a candidate relates to.

Finally, we link each f to the highest ranking candidate meaning v^* if $\text{score}((v^*, f)) \geq \theta$, where θ is a fixed threshold (see lines 14–18 of Algorithm 2). For instance, in sentence (1) and for the fragment *Mario* we select *Mario Gomez* as our final candidate meaning and link it to the fragment.

Linking by densest subgraph. We now illustrate our novel densest subgraph heuristic, used in line 12 of Algorithm 2, for reducing the level of ambiguity of the initial semantic interpretation graph G_I . The main idea here is that the most suitable meanings of each text fragment will belong to the densest area of the graph. For instance, in Figure 1 the (candidate, fragment) pairs *(Thomas Müller, Thomas)*, *(Mario Gomez, Mario)*, *(striker, striker)* and *(FC Bayern Munich, Munich)* form a dense subgraph supporting their relevance for sentence (1).

⁵We denote with $\text{deg}(v)$ the overall number of incoming and outgoing edges, i.e., $\text{deg}(v) := \text{deg}^+(v) + \text{deg}^-(v)$.

Algorithm 2 Candidate Disambiguation.

- 1: **input:** F , the fragments in the input text;
 semSign , the semantic signatures;
 μ , ambiguity level to be reached;
 cand , fragments to candidate meanings.
 - 2: **output:** *selected*, disambiguated fragments.
 - 3: **function** DISAMB($F, \text{semSign}, \mu, \text{cand}$)
 - 4: $V_I := \emptyset; E_I := \emptyset$
 - 5: $G_I := (V_I, E_I)$
 - 6: **for each** fragment $f \in F$ **do**
 - 7: **for each** candidate $v \in \text{cand}(f)$ **do**
 - 8: $V_I := V_I \cup \{(v, f)\}$
 - 9: **for each** $((v, f), (v', f')) \in V_I \times V_I$ **do**
 - 10: **if** $f \neq f'$ **and** $v' \in \text{semSign}_v$ **then**
 - 11: $E_I := E_I \cup \{((v, f), (v', f'))\}$
 - 12: $G_I^* := \text{DENSSUB}(F, \text{cand}, G_I, \mu)$
 - 13: *selected* := **new** Map < String, Synset >
 - 14: **for each** $f \in F$ s.t. $\exists (v, f) \in V_I^*$ **do**
 - 15: $\text{cand}^*(f) := \{v : (v, f) \in V_I^*\}$
 - 16: $v^* := \arg \max_{v \in \text{cand}^*(f)} \text{score}((v, f))$
 - 17: **if** $\text{score}((v^*, f)) \geq \theta$ **then**
 - 18: $\text{selected}(f) := v^*$
 - 19: **return** *selected*
-

Munich, Munich) form a dense subgraph supporting their relevance for sentence (1).

The problem of identifying the densest subgraph of size at least k is NP-hard (Feige et al., 1999). Therefore, we define a heuristic for k -partite graphs inspired by a 2-approximation greedy algorithm for arbitrary graphs (Charikar, 2000; Khuller and Saha, 2009). Our adapted strategy for selecting a dense subgraph of G_I is based on the iterative removal of low-coherence vertices, i.e., fragment interpretations. We show the pseudocode in Algorithm 3.

We start with the initial graph $G_I^{(0)}$ at step $t = 0$ (see line 5). For each step t (lines 7–16), first, we identify the most ambiguous fragment f_{\max} , i.e., the one with the maximum number of candidate mean-

Algorithm 3 Densest Subgraph.

```
1: input:  $F$ , the set of all fragments in the input text;  
    $cand$ , from fragments to candidate meanings;  
    $G_I^{(0)}$ , the full semantic interpretation graph;  
    $\mu$ , ambiguity level to be reached.  
2: output:  $G_I^*$ , a dense subgraph.  
3: function DENSUB( $F, cand, G_I^{(0)}, \mu$ )  
4:    $t := 0$   
5:    $G_I^* := G_I^{(0)}$   
6:   while true do  
7:      $f_{max} := \arg \max_{f \in F} |\{v : \exists(v, f) \in V_I^{(t)}\}|$   
8:     if  $|\{v : \exists(v, f_{max}) \in V_I^{(t)}\}| \leq \mu$  then  
9:       break;  
10:     $v_{min} := \underset{v \in cand(f_{max})}{\operatorname{argmin}} \operatorname{score}((v, f_{max}))$   
11:     $V_I^{(t+1)} := V_I^{(t)} \setminus \{(v_{min}, f_{max})\}$   
12:     $E_I^{(t+1)} := E_I^{(t)} \cap V_I^{(t+1)} \times V_I^{(t+1)}$   
13:     $G_I^{(t+1)} := (V_I^{(t+1)}, E_I^{(t+1)})$   
14:    if  $\operatorname{avgdeg}(G_I^{(t+1)}) > \operatorname{avgdeg}(G_I^*)$  then  
15:       $G_I^* := G_I^{(t+1)}$   
16:     $t := t + 1$   
17:  return  $G_I^*$ 
```

ings in the graph (see line 7). Next, we discard the weakest interpretation of the current fragment f_{max} . To do so, we determine the lexical and semantic coherence of each candidate meaning (v, f_{max}) using Formula 2 (see line 10). We then remove from our graph $G_I^{(t)}$ the lowest-coherence vertex (v_{min}, f_{max}) , i.e., the one whose score is minimum (see lines 11–13). For instance, in Figure 1, f_{max} is the fragment *Mario* and we have: $\operatorname{score}((\text{Mario Gomez}, \text{Mario})) \propto \frac{3}{3} \cdot 5 = 5$, $\operatorname{score}((\text{Mario Basler}, \text{Mario})) \propto \frac{1}{3} \cdot 1 = 0.\bar{3}$ and $\operatorname{score}((\text{Mario Adorf}, \text{Mario})) \propto \frac{2}{3} \cdot 2 = 1.\bar{3}$, so we remove $(\text{Mario Basler}, \text{Mario})$ from the graph since its score is minimum.

We then move to the next step, i.e., we set $t := t + 1$ (see line 16) and repeat the low-coherence removal step. We stop when the number of remaining candidates for each fragment is below a threshold μ , i.e., $|\{v : \exists(v, f) \in V_I^{(t)}\}| \leq \mu \forall f \in F$ (see lines 8–9). During each iteration step t we compute the average degree of the current graph $G_I^{(t)}$, i.e., $\operatorname{avgdeg}(G_I^{(t)}) = \frac{2|E_I^{(t)}|}{|V_I^{(t)}|}$. Finally, we select as the densest subgraph of the initial semantic interpretation graph G_I the graph G_I^* that maximizes the average degree (see lines 14–15).

8 Experimental Setup

Datasets. We carried out our experiments on six datasets, four for WSD and two for EL:

- The SemEval-2013 task 12 dataset for multilingual WSD (Navigli et al., 2013), which consists of 13 documents in different domains, available in 5 languages. For each language, all noun occurrences were annotated using BabelNet, thereby providing Wikipedia and WordNet annotations wherever applicable. The number of mentions to be disambiguated roughly ranges from 1K to 2K per language in the different setups.
- The SemEval-2007 task 7 dataset for coarse-grained English all-words WSD (Navigli et al., 2007). We take into account only nominal mentions obtaining a dataset containing 1107 nouns to be disambiguated using WordNet.
- The SemEval-2007 task 17 dataset for fine-grained English all-words WSD (Pradhan et al., 2007). We considered only nominal mentions resulting in 158 nouns annotated with WordNet synsets.
- The Senseval-3 dataset for English all-words WSD (Snyder and Palmer, 2004), which contains 899 nouns to be disambiguated using WordNet.
- KORE50 (Hoffart et al., 2012), which consists of 50 short English sentences (mean length of 14 words) with a total number of 144 mentions manually annotated using YAGO2, for which a Wikipedia mapping is available. This dataset was built with the idea of testing against a high level of ambiguity for the EL task.
- AIDA-CoNLL⁶ (Hoffart et al., 2011), which consists of 1392 English articles, for a total of roughly 35K named entity mentions annotated with YAGO concepts separated in development, training and test sets.

We exploited the POS tags already available in the SemEval and Senseval datasets, while we used the Stanford POS tagger (Toutanova et al., 2003) for the English sentences in the last two datasets.

⁶We used AIDA-CoNLL as it is the most recent and largest available dataset for EL (Hachey et al., 2013). The TAC KBP datasets are available only to participants.

Parameters. We fixed the parameters of RWR (Section 5) to the values $\alpha = .85$, $\eta = 100$ and $n = 1M$ which maximize F1 on a manually created tuning set made up of 10 gold-standard semantic signatures. We tuned our two disambiguation parameters $\mu = 10$ and $\theta = 0.8$ by optimizing $F1$ on the trial dataset of the SemEval-2013 task on multilingual WSD (Navigli et al., 2013). We used the same parameters on all the other WSD datasets. As for EL, we used the training part of AIDA-CoNLL (Hoffart et al., 2011) to set $\mu = 5$ and $\theta = 0.0$.

8.1 Systems

Multilingual WSD. We evaluated our system on the SemEval-2013 task 12 by comparing it with the participating systems:

- UMCC-DLSI (Gutiérrez et al., 2013) a state-of-the-art Personalized PageRank-based approach that exploits the integration of different sources of knowledge, such as WordNet Domains/Affect (Strapparava and Valitutti, 2004), SUMO (Zouaq et al., 2009) and the eXtended WordNet (Mihalcea and Moldovan, 2001);
- DAEBAK! (Manion and Sainudiin, 2013) which performs WSD on the basis of peripheral diversity within subgraphs of BabelNet;
- GETALP (Schwab et al., 2013) which uses an Ant Colony Optimization technique together with the classical measure of Lesk (1986).

We also compared with UKB w2w (Agirre and Soroa, 2009), a state-of-the-art approach for knowledge-based WSD, based on Personalized PageRank (Haveliwala, 2002). We used the same mapping from words to senses that we used in our approach, default parameters⁷ and BabelNet as the input graph. Moreover, we compared our system with IMS (Zhong and Ng, 2010), a state-of-the-art supervised English WSD system which uses an SVM trained on sense-annotated corpora, such as SemCor (Miller et al., 1993) and DSO (Ng and Lee, 1996), among others. We used the IMS model out-of-the-box with Most Frequent Sense (MFS) as backoff routine since the model obtained using the task trial data performed worse.

We followed the original task formulation and evaluated the synsets in three different settings, i.e.,

⁷`./ukb_wsd -D dict.txt -K kb.bin --ppr_w2w ctx.txt`

when using BabelNet senses, Wikipedia senses and WordNet senses, thanks to BabelNet being a superset of the other two inventories. We ran our system on a document-by-document basis, i.e., disambiguating each document at once, so as to test its effectiveness on long coherent texts. Performance was calculated in terms of F1 score. We also compared the systems with the MFS baseline computed for the three inventories (Navigli et al., 2013).

Coarse-grained WSD. For the SemEval-2007 task 7 we compared our system with the two top-ranked approaches, i.e., NUS-PT (Chan et al., 2007) and UoR-SSI (Navigli, 2008), which respectively exploited parallel texts and enriched semantic paths in a semantic network, the previously described UKB w2w system,⁸ a knowledge-based WSD approach (Ponzetto and Navigli, 2010) which exploits an automatic extension of WordNet, and, as baseline, the MFS.

Fine-grained WSD. For the remaining fine-grained WSD datasets, i.e., Senseval-3 and SemEval-2007 task 17, we compared our approach with the previously described state-of-the-art systems UKB and IMS, and, as baseline, the MFS.

KORE50 and AIDA-CoNLL. For the KORE50 and AIDA-CoNLL datasets we compared our system with six approaches, including state-of-the-art ones (Hoffart et al., 2012; Cornolti et al., 2013):

- MW, i.e., the Normalized Google Distance as defined by Milne and Witten (2008);
- KPCS (Hoffart et al., 2012), which calculates a Mutual Information weighted vector of keyphrases for each candidate and then uses the cosine similarity to obtain candidates' scores;
- KORE and its variants $KORE_{LSH-G}$ and $KORE_{LSH-F}$ (Hoffart et al., 2012), based on similarity measures that exploit the overlap between phrases associated with the considered entities (KORE) and a hashing technique to reduce the space needed by the keyphrases associated with the entities (LSH-G, LSH-F);
- Tagme 2.0⁹ (Ferragina and Scaiella, 2012) which uses the relatedness measure defined

⁸We report the results as given by Agirre et al. (2014).

⁹We used the out-of-the-box RESTful API available at <http://tagme.di.unipi.it>

System	Sens3		Sem07			SemEval-2013 English			French		German		Italian		Spanish	
	WN	WN	WN	Wiki	BN	Wiki	BN	Wiki	BN	Wiki	BN	Wiki	BN	Wiki	BN	
Babelfy	68.3	62.7	65.9	87.4	69.2	71.6	*56.9	81.6	69.4	84.3	66.6	83.8	69.5			
IMS	71.2	63.3	65.7	–	–	–	–	–	–	–	–	–	–	–	–	
UKB w2w	*65.3	*56.0	61.3	–	60.8	–	60.8	–	66.2	–	67.3	–	70.0			
UMCC-DLSI	–	–	64.7	54.8	68.5	*60.5	60.5	*58.1	62.8	*58.3	65.8	*61.0	71.0			
DAEBAK!	–	–	–	–	60.4	–	53.8	–	59.1	–	*61.3	–	60.0			
GETALP-BN	–	–	51.4	–	58.3	–	48.3	–	52.3	–	52.8	–	57.8			
MFS	70.3	65.8	*63.0	*80.3	*66.5	69.4	45.3	83.1	*67.4	82.3	57.5	82.4	*64.4			
Babelfy unif. weights	67.0	65.2	65.0	87.0	68.5	71.9	57.2	81.2	69.8	83.7	66.8	83.8	70.8			
Babelfy w/o dens. sub.	68.3	63.3	65.4	87.3	68.7	71.6	57.0	81.7	69.1	84.4	66.5	83.9	69.5			
Babelfy only concepts	68.2	62.7	65.5	83.0	68.7	70.2	56.6	79.3	69.3	83.0	66.3	84.0	69.7			
Babelfy on sentences	66.0	65.2	63.5	84.0	67.1	70.7	53.6	82.3	68.1	83.8	64.2	83.5	68.7			

Table 1: F1 scores (percentages) of the participating systems of SemEval-2013 task 12 together with MFS, UKB w2w, IMS, our system and its ablated versions on the Senseval-3, SemEval-2007 task 17 and SemEval-2013 datasets. The first system which has a statistically significant difference from the top system is marked with * ($\chi^2, p < 0.05$).

by Milne and Witten (2008) weighted with the commonness of a sense together with the keyphraseness measure defined by Mihalcea and Csomai (2007) to exploit the context around the target word;

- Illinois Wikifier¹⁰ (Cheng and Roth, 2013) which combines local features, such as commonness and TF-IDF between mentions and Wikipedia pages, with global coherence features based on Wikipedia links and relational inference;
- DBpedia Spotlight¹¹ (Mendes et al., 2011) which uses LingPipe’s string matching algorithm implementation together with a weighted cosine similarity measure to recognize and disambiguate mentions.

We also compared with UKB w2w, introduced above. Note that we could not use supervised systems, as the training data of AIDA-CoNLL covers less than half of the mentions used in the testing part and less than 10% of the entities considered in KORE50. To enable a fair comparison, we ran our system by restricting the BabelNet sense inventory of the target mentions to the English Wikipedia. As is customary in the literature, we calculated the systems’ accuracy for both Entity Linking datasets.

¹⁰We used the out-of-the-box Java API available from http://cogcomp.cs.illinois.edu/page/download_view/Wikifier

¹¹We used the 2011 version of DBpedia Spotlight as it obtains better scores on the considered datasets in comparison to the new version (Daiber et al., 2013). We used the out-of-the-box RESTful API available at <http://spotlight.dbpedia.org>

9 Results

Multilingual WSD. In Table 1 we show the F1 performance on the SemEval-2013 task 12 for the three setups: WordNet, Wikipedia and BabelNet. Using BabelNet we surpass all systems on English and German and obtain performance comparable with the best systems on two other languages (UKB on Italian and UMCC-DLSI on Spanish). Using the WordNet sense inventory, our results are on a par with the best system, i.e., IMS. On Wikipedia our results range between 71.6% (French) and 87.4% F1 (English), i.e., more than 10 points higher than the current state of the art (UMCC-DLSI) in all 5 languages. As for the MFS baseline, which is known to be very competitive in WSD (Navigli, 2009), we beat it in all setups except for German on Wikipedia. Interestingly, we surpass the WordNet MFS by 2.9 points, a significant result for a knowledge-based system (see also (Pilehvar and Navigli, 2014)).

Coarse- and fine-grained WSD. In Table 2, we show the results of the systems on the SemEval-2007 coarse-grained WSD dataset. As can be seen, we obtain the second best result after Ponzetto and Navigli (2010). In Table 1 (first two columns), we show the results of IMS and UKB on the Senseval-3 and SemEval-2007 task 17 datasets. We rank second on both datasets after IMS. However, the differences are not statistically significant. Moreover, Agirre et al. (2014, Table 5) note that using WordNet 3.0, instead of 1.7 or 2.1, to annotate these datasets can cause a more than one percent drop in performance.

System	F1
(Ponzetto and Navigli, 2010)	85.5
Babelfy	84.6
UoR-SSI	84.1
UKB w2w	83.6
NUS-PT	*82.3
MFS	77.4
Babelfy unif. weights	85.7
Babelfy w/o dens. sub.	84.9
Babelfy only concepts	85.3
Babelfy on sentences	82.3

Table 2: F1 score (percentages) on the SemEval-2007 task 7. The first system which has a statistically significant difference from the top system is marked with \star (χ^2 , $p < 0.05$).

Entity Linking. In Table 3 we show the results on the two Entity Linking datasets, i.e., KORE50 and AIDA-CoNLL. Our system outperforms all other approaches, with KORE-LSH-G getting closest, and Tagme and Wikifier lagging behind on the KORE50 dataset. For the AIDA-CoNLL dataset we obtain the third best performance after MW and KPCS, however the difference is not statistically significant.

We note the low performance of DBpedia Spotlight which, even if it achieves almost 100% precision on the identified mentions on both datasets, suffers from low recall due to its candidate identification step, confirming previous evaluations (Derczynski et al., 2013; Hakimov et al., 2012; Ludwig and Sack, 2011). This problem becomes even more accentuated in the latest version of this system (Daiber et al., 2013). Finally, UKB using BabelNet obtains low performance on EL, i.e., 19.4-10.5 points below the state of the art. This result is discussed below.

Discussion. The results obtained by UKB show that the high performance of our unified approach to EL and WSD is not just a mere artifact of the use of a rich multilingual semantic network, that is, BabelNet. In other words, it is not true that any graph-based algorithm could be applied to perform both EL and WSD at the same time equally well. This also shows that BabelNet by itself is not sufficient for achieving high performances for both tasks and that, instead, an appropriate processing of the structural and lexical information of the semantic network is needed. A manual analysis revealed that the main cause of error for UKB in the EL setup stems

System	KORE50	CoNLL
Babelfy	71.5	82.1
KORE-LSH-G	64.6	81.8
KORE	63.9	*80.7
MW	*57.6	82.3
Tagme	56.3	70.1
KPCS	55.6	82.2
KORE-LSH-F	53.2	81.2
UKB w2w (on BabelNet)	52.1	71.8
Illinois Wikifier	41.7	72.4
DBpedia Spotlight	35.4	34.0
Babelfy unif. weights	69.4	81.7
Babelfy w/o dens. sub.	62.5	78.1
Babelfy only NE	68.1	78.8

Table 3: Accuracy (percentages) of state-of-the-art EL systems and our system on KORE50 and AIDA-CoNLL. The first system with a statistically significant difference from the top system is marked with \star (χ^2 , $p < 0.05$).

from its inability to enforce high coherence, e.g., by jointly disambiguating all the words, which is instead needed when considering the high level of ambiguity that we have in our semantic interpretation graph (Cucerzan, 2007). For instance, for sentence (1) in the introduction, UKB disambiguates *Thomas* as a cricket player and *Mario* as the popular video game rather than the two well-known soccer players, and *Munich* as the German city, rather than the soccer team in which they play. Our approach, instead, by enforcing highly coherent semantic interpretations, correctly identifies all the soccer-related entities.

In order to determine the need of our loose candidate identification heuristic (see Section 6), we compared the percentage of times a candidate set contains the correct entity against that obtained by an exact string matching between the mention and the sense inventory. On KORE50, our heuristic retrieves the correct entity 98.6% of the time vs. 42.4% when exact matching is used. This demonstrates the inadequacy of exact matching for EL, and the need for a comprehensive sense inventory, as is done in our approach.

We also performed different ablation tests by experimenting with the following variants of our system (reported at the bottom of Tables 1, 2 and 3):

- Babelfy using uniform distribution during the RWR to obtain the concepts’ semantic signatures; this test assesses the impact of our weighting and edge creation strategy.

- Babelfy without performing the densest subgraph heuristic, i.e., when line 12 in Algorithm 2 is $G_I^* = G_I$, so as to verify the impact of identifying the most coherent interpretations.
- Babelfy applied to the BabelNet subgraph induced by the entire set of named entity vertices, for the EL task, and that induced by word senses only, for the WSD task; this test aims to stress the impact of our unified approach.
- Babelfy applied on sentences instead of on whole documents.

The component which has a smaller impact on the performance is our triangle-based weighting scheme. The main exception is on the smallest dataset, i.e., SemEval-2007 task 17, for which this version attains an improvement of 2.5 percentage points.

Babelfy without the densest subgraph algorithm is the version which attains the lowest performances on the EL task, with a 9% performance drop on the KORE50 dataset, showing the need for a specially designed approach to cope with the high level of ambiguity that is encountered on this task. On the other hand, in the WSD datasets this version attains almost the same results as the full version, due to the lower number of candidate word senses.

Babelfy applied on sentences instead of on whole documents shows a lower performance, confirming the significance of higher semantic coherence on whole documents (notwithstanding the two exceptions on the SemEval-2007 task 17 and on the SemEval-2013 German Wikipedia datasets).

Finally, the version in which we restrict our system to named entities only (for EL) and concepts only (for WSD) consistently obtains lower results (notwithstanding the three exceptions on the Spanish SemEval-2013 task 12 using BabelNet and Wikipedia, and on the SemEval 2007 coarse-grained task). This highlights the benefit of our joint use of lexicographic and encyclopedic structured knowledge, on each of the two tasks. The 3.4% performance drop attained on KORE50 is of particular interest, since this dataset aims at testing performance on highly ambiguous mentions within short sentences. This indicates that the semantic analysis of small contexts can be improved by leveraging the coherence between concepts and named entities.

10 Conclusion

In this paper we presented Babelfy, a novel, integrated approach to Entity Linking and Word Sense Disambiguation, available at <http://babelfy.org>. Our joint solution is based on three key steps: i) the automatic creation of semantic signatures, i.e., related concepts and named entities, for each node in the reference semantic network; ii) the unconstrained identification of candidate meanings for all possible textual fragments; iii) linking based on a high-coherence densest subgraph algorithm. We used BabelNet 1.1.1 as our multilingual semantic network.

Our graph-based approach exploits the semantic network structure to its advantage: two key features of BabelNet, that is, its multilinguality and its integration of lexicographic and encyclopedic knowledge, make it possible to run our general, unified approach on the two tasks of Entity Linking and WSD in any of the languages covered by the semantic network. However, we also demonstrated that BabelNet in itself does not lead to state-of-the-art accuracy on both tasks, even when used in conjunction with a high-performance graph-based algorithm like Personalized PageRank. This shows the need for our novel unified approach to EL and WSD.

At the core of our approach lies the effective treatment of the high degree of ambiguity of partial textual mentions by means of a 2-approximation algorithm for the densest subgraph problem, which enables us to output a semantic interpretation of the input text with drastically reduced ambiguity, as was previously done with SSI (Navigli, 2008).

Our experiments on six gold-standard datasets show the state-of-the-art performance of our approach, as well as its robustness across languages. Our evaluation also demonstrates that our approach fares well both on long texts, such as those of the WSD tasks, and short and highly-ambiguous sentences, such as the ones in KORE50. Finally, ablation tests and further analysis demonstrate that each component of our system is needed to contribute state-of-the-art performances on both EL and WSD.

As future work, we plan to use Babelfy for information extraction, where semantics is taking the lead (Moro and Navigli, 2013), and for the validation of semantic annotations (Vannella et al., 2014).

Acknowledgments



The authors gratefully acknowledge the support of the ERC Starting Grant MultiJEDI No. 259234.



References

- Eneko Agirre and Aitor Soroa. 2009. Personalizing PageRank for Word Sense Disambiguation. In *Proc. of EACL*, pages 33–41.
- Eneko Agirre, David Martínez, Oier López de Lacalle, and Aitor Soroa. 2006. Two graph-based algorithms for state-of-the-art WSD. In *Proc. of EMNLP*, pages 585–593.
- Eneko Agirre, Aitor Soroa, and Mark Stevenson. 2010. Graph-based Word Sense Disambiguation of biomedical documents. *Bioinformatics*, 26(22):2889–2896.
- Eneko Agirre, Oier Lopez de Lacalle, and Aitor Soroa. 2014. Random Walks for Knowledge-Based Word Sense Disambiguation. *Computational Linguistics*, 40(1):57–84.
- Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A Nucleus for a Web of Open Data. In *Proc. of ISWC/ASWC*, pages 722–735.
- Carmen Banea and Rada Mihalcea. 2011. Word Sense Disambiguation with multilingual features. In *Proc. of IWCS*, pages 25–34.
- Christoph Böhm, Gerard de Melo, Felix Naumann, and Gerhard Weikum. 2012. LINDA: distributed web-of-data-scale entity matching. In *Proc. of CIKM*, pages 2104–2108.
- Samuel Brody and Mirella Lapata. 2009. Bayesian Word Sense Induction. In *Proc. of EACL*, pages 103–111.
- Razvan C. Bunescu and Marius Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proc. of EACL*, pages 9–16.
- Yee Seng Chan, Hwee Tou Ng, and Zhi Zhong. 2007. NUS-PT: Exploiting Parallel Texts for Word Sense Disambiguation in the English All-Words Tasks. In *Proc. of SemEval-2007*, pages 253–256.
- Moses Charikar. 2000. Greedy approximation algorithms for finding dense components in a graph. In *Proc. of APPROX*, pages 84–95.
- Xiao Cheng and Dan Roth. 2013. Relational Inference for Wikification. In *Proc. of EMNLP*, pages 1787–1796.
- Marco Cornolti, Paolo Ferragina, and Massimiliano Ciaramita. 2013. A framework for benchmarking entity-annotation systems. In *Proc. of WWW*, pages 249–260.
- Silviu Cucerzan. 2007. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. In *Proc. of EMNLP-CoNLL*, pages 708–716.
- Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. 2013. Improving efficiency and accuracy in multilingual entity extraction. In *Proc. of I-Semantics*, pages 121–124.
- Leon Derczynski, Diana Maynard, Niraj Aswani, and Kalina Bontcheva. 2013. Microblog-genre noise and impact on semantic annotation accuracy. In *Proc. of Hypertext*, pages 21–30.
- Antonio Di Marco and Roberto Navigli. 2013. Clustering and Diversifying Web Search Results with Graph-Based Word Sense Induction. *Computational Linguistics*, 39(3):709–754.
- Nicolai Erbs, Torsten Zesch, and Iryna Gurevych. 2011. Link discovery: A comprehensive analysis. In *Proc. of ICSC*, pages 83–86.
- Oren Etzioni, Michele Banko, and Michael J Cafarella. 2006. Machine Reading. In *Proc. of AAAI*, pages 1517–1519.
- Uriel Feige, Guy Kortsarz, and David Peleg. 1999. The dense k-subgraph problem. *Algorithmica*, 29:2001.
- Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.
- Paolo Ferragina and Ugo Scaiella. 2010. TAGME: On-the-fly Annotation of Short Text Fragments (by Wikipedia Entities). In *Proc. of CIKM*, pages 1625–1628.
- Paolo Ferragina and Ugo Scaiella. 2012. Fast and Accurate Annotation of Short Texts with Wikipedia Pages. *IEEE Software*, 29(1):70–75.
- Weiwei Guo and Mona T. Diab. 2010. Combining Orthogonal Monolingual and Multilingual Sources of Evidence for All Words WSD. In *Proc. of ACL*, pages 1542–1551.
- Yoan Gutiérrez, Yenier Castañeda, Andy González, Rainel Estrada, Dennys D. Piug, Jose I. Abreu, Roger Pérez, Antonio Fernández Orquín, Andrés Montoyo, Rafael Muñoz, and Franc Camara. 2013. UMCC.DLSI: Reinforcing a Ranking Algorithm with Sense Frequencies and Multidimensional Semantic Resources to solve Multilingual Word Sense Disambiguation. In *Proc. of SemEval-2013*, pages 241–249.
- Ben Hachey, Will Radford, Joel Nothman, Matthew Honnibal, and James R. Curran. 2013. Evaluating Entity Linking with Wikipedia. *Artificial Intelligence*, 194:130–150.
- Sherzod Hakimov, Salih Atilay Oto, and Erdogan Dogdu. 2012. Named entity recognition and disambiguation using linked data and graph-based centrality scoring. In *Proc. of SWIM*, pages 4:1–4:7.
- Taher H. Haveliwala. 2002. Topic-sensitive PageRank. In *Proc. of WWW*, pages 517–526.

- Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenu, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proc. of EMNLP*, pages 782–792.
- Johannes Hoffart, Stephan Seufert, Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. 2012. KORE: keyphrase overlap relatedness for entity disambiguation. In *Proc. of CIKM*, pages 545–554.
- Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence*, 194:28–61.
- Eduard H. Hovy, Roberto Navigli, and Simone P. Ponzetto. 2013. Collaboratively built semi-structured content and Artificial Intelligence: The story so far. *Artificial Intelligence*, 194:2–27.
- Samir Khuller and Barna Saha. 2009. On finding dense subgraphs. In *Proc. of ICALP*, pages 597–608.
- Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. 2009. Collective Annotation of Wikipedia Entities in Web Text. In *Proc. of KDD*, pages 457–466.
- Els Lefever and Véronique Hoste. 2010. Semeval-2010 task 3: Cross-lingual Word Sense Disambiguation. In *Proc. of SemEval-2010*, pages 15–20.
- Els Lefever and Véronique Hoste. 2013. SemEval-2013 Task 10: Cross-lingual Word Sense Disambiguation. In *Proc. of SemEval-2013*, pages 158–166.
- Els Lefever, Véronique Hoste, and Martine De Cock. 2011. Parasense or how to use parallel corpora for Word Sense Disambiguation. In *Proc. of ACL-HLT*, pages 317–322.
- Michael E. Lesk. 1986. Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone. In *Proc. of the International Conference on Systems Documentation*, pages 24–26.
- Nadine Ludwig and Harald Sack. 2011. Named entity recognition for user-generated tags. In *Proc. of DEXA*, pages 177–181.
- Suresh Manandhar, Ioannis P. Klapaftis, Dmitriy Dligach, and Sameer S. Pradhan. 2010. SemEval-2010 task 14: Word sense induction & disambiguation. In *Proc. of SemEval-2010*, pages 63–68.
- Steve L. Manion and Raazesh Sainudiin. 2013. DAE-BAK!: Peripheral Diversity for Multilingual Word Sense Disambiguation. In *Proc. of SemEval-2013*, pages 250–254.
- Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. 2011. DBpedia spotlight: shedding light on the web of documents. In *Proc. of I-Semantics*, pages 1–8.
- Rada Mihalcea and Andras Csomai. 2007. Wikify!: linking documents to encyclopedic knowledge. In *Proc. of CIKM*, pages 233–242.
- Rada Mihalcea and Dan I Moldovan. 2001. Extended WordNet: Progress report. In *Proc. of NAACL Workshop on WordNet and Other Lexical Resources*, pages 95–100.
- Rada Mihalcea. 2005. Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling. In *Proc. of HLT/EMNLP*, pages 411–418.
- George A. Miller, Claudia Leacock, Randee Teng, and Ross T. Bunker. 1993. A semantic concordance. In *Proc. of HLT*, pages 303–308.
- Tristan Miller, Chris Biemann, Torsten Zesch, and Iryna Gurevych. 2012. Using Distributional Similarity for Lexical Expansion in Knowledge-based Word Sense Disambiguation. In *Proc. of COLING*, pages 1781–1796.
- David Milne and Ian H. Witten. 2008. Learning to link with Wikipedia. In *Proc. of CIKM*, pages 509–518.
- Andrea Moro and Roberto Navigli. 2013. Integrating Syntactic and Semantic Analysis into the Open Information Extraction Paradigm. In *Proc. of IJCAI*, pages 2148–2154.
- Roberto Navigli and Mirella Lapata. 2010. An Experimental Study of Graph Connectivity for Unsupervised Word Sense Disambiguation. *TPAMI*, 32(4):678–692.
- Roberto Navigli and Simone Paolo Ponzetto. 2012a. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.
- Roberto Navigli and Simone Paolo Ponzetto. 2012b. Joining forces pays off: Multilingual Joint Word Sense Disambiguation. In *Proc. of EMNLP*, pages 1399–1410.
- Roberto Navigli, Kenneth C. Litkowski, and Orin Hargraves. 2007. SemEval-2007 Task 07: Coarse-Grained English All-Words Task. In *Proc. of SemEval-2007*, pages 30–35.
- Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. SemEval-2013 Task 12: Multilingual Word Sense Disambiguation. In *Proc. of SemEval-2013*, pages 222–231.
- Roberto Navigli. 2008. A structural approach to the automatic adjudication of word sense disagreements. *Natural Language Engineering*, 14(4):293–310.
- Roberto Navigli. 2009. Word Sense Disambiguation: A survey. *ACM Computing Surveys*, 41(2):1–69.
- Roberto Navigli. 2012. A Quick Tour of Word Sense Disambiguation, Induction and Related Approaches. In *Proc. of SOFSEM*, pages 115–129.

- Hwee Tou Ng and Hian Beng Lee. 1996. Integrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach. In *Proc. of ACL*, pages 40–47.
- Mohammad Taher Pilehvar and Roberto Navigli. 2014. A Large-scale Pseudoword-based Evaluation Framework for State-of-the-Art Word Sense Disambiguation. *Computational Linguistics*.
- Simone P. Ponzetto and Roberto Navigli. 2010. Knowledge-rich Word Sense Disambiguation rivaling supervised system. In *Proc. of ACL*, pages 1522–1531.
- Sameer S. 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*, pages 87–92. Association for Computational Linguistics.
- Delip Rao, Paul McNamee, and Mark Dredze. 2013. Entity Linking: Finding Extracted Entities in a Knowledge Base. In *Multi-source, Multilingual Information Extraction and Summarization, Theory and Applications of Natural Language Processing*, pages 93–115. Springer Berlin Heidelberg.
- Lev-Arie Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and Global Algorithms for Disambiguation to Wikipedia. In *Proc. of ACL*, pages 1375–1384.
- Lenhart K. Schubert. 2006. Turing’s dream and the knowledge challenge. In *Proc. of NCAI*, pages 1534–1538.
- Didier Schwab, Andon Tchechmedjiev, Jérôme Goulian, Mohammad Nasiruddin, Gilles Sérasset, and Hervé Blanchon. 2013. GETALP System: Propagation of a Lesk Measure through an Ant Colony Algorithm. In *Proc. of SemEval-2013*, pages 232–240.
- Hui Shen, Razvan Bunescu, and Rada Mihalcea. 2013. Coarse to Fine Grained Sense Disambiguation in Wikipedia. In *Proc. of *SEM*, pages 22–31.
- Ravi Sinha and Rada Mihalcea. 2007. Unsupervised Graph-based Word Sense Disambiguation Using Measures of Word Semantic Similarity. In *Proc. of ICSC*, pages 363–369.
- Benjamin Snyder and Martha Palmer. 2004. The English all-words task. In *Proc. of Senseval-3*, pages 41–43.
- Carlo Strapparava and Alessandro Valitutti. 2004. WordNet Affect: an Affective Extension of WordNet. In *Proc. of LREC*, pages 1083–1086.
- Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2006. Fast Random Walk with Restart and Its Applications. In *Proc. of ICDM*, pages 613–622.
- Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. of NAACL-HLT*, pages 173–180.
- George Tsatsaronis, Michalis Vazirgiannis, and Ion Androutsopoulos. 2007. Word Sense Disambiguation with Spreading Activation Networks Generated from Thesauri. In *Proc. of IJCAI*, pages 1725–1730.
- Tim Van de Cruys and Marianna Apidianaki. 2011. Latent Semantic Word Sense Induction and Disambiguation. In *Proc. of ACL*, pages 1476–1485.
- Daniele Vannella, David Jurgens, Daniele Scarfini, Domenico Toscani, and Roberto Navigli. 2014. Validating and Extending Semantic Knowledge Bases using Video Games with a Purpose. In *Proc. of ACL*.
- Duncan J. Watts and Steven H. Strogatz. 1998. Collective dynamics of ‘small-world’ networks. *Nature*, 393(6684):409–10.
- Zhi Zhong and Hwee Tou Ng. 2010. It Makes Sense: A Wide-Coverage Word Sense Disambiguation System for Free Text. In *Proc. of ACL (Demo)*, pages 78–83.
- Amal Zouaq, Michel Gagnon, and Benoit Ozell. 2009. A SUMO-based Semantic Analysis for Knowledge Extraction. In *Proc of LTC*.

Data-Driven Metaphor Recognition and Explanation

Hongsong Li

Microsoft Research Asia
hongsli@microsoft.com

Kenny Q. Zhu

Shanghai Jiao Tong University
kzhu@cs.sjtu.edu.cn

Haixun Wang

Google Research
haixun@google.com

Abstract

Recognizing metaphors and identifying the source-target mappings is an important task as metaphorical text poses a big challenge for machine reading. To address this problem, we automatically acquire a metaphor knowledge base and an isA knowledge base from billions of web pages. Using the knowledge bases, we develop an inference mechanism to recognize and explain the metaphors in the text. To our knowledge, this is the first purely data-driven approach of probabilistic metaphor acquisition, recognition, and explanation. Our results show that it significantly outperforms other state-of-the-art methods in recognizing and explaining metaphors.

1 Introduction

A metaphor is a way of communicating. It enables us to comprehend one thing in terms of another. For example, the metaphor, *Juliet is the sun*, allows us to see Juliet much more vividly than if Shakespeare had taken a more literal approach. We utter about one metaphor for every ten to twenty-five words, or about six metaphors a minute (Geary, 2011).

Specifically, a metaphor is a mapping of concepts from a source domain to a target domain (Lakoff and Johnson, 1980). The source domain is often concrete and based on sensory experience, while target domain is usually abstract. Two concepts are connected by this mapping because they share some common or similar properties, and as a result, the meaning of one concept can be *transferred* to another. For example, in “Juliet is the sun,” *the sun* is the source concept while *Juliet* is the target concept.

One interpretation of this metaphor is that both concepts share the property that their existence brings about warmth, life, and excitement. In a metaphorical sentence, at least one of the two concepts must be explicitly present. This leads to three types of metaphors:

1. *Juliet is the sun*. Here, both the source (*sun*) and the target (*Juliet*) are explicit.
2. *Please wash your claws before scratching me*. Here, the source (*claws*) is explicit, while the target (*hands*) is implicit, and the context of *wash* is in terms of the target.
3. *Your words cut deep*. Here, the target (*words*) is explicit, while the source (possibly, *knife*) is implicit, and the context of *cut* is in terms of the source.

In this paper, we focus on the *recognition* and *explanation* of metaphors. For a given sentence, we first check whether it contains a metaphoric expression (which we call metaphor recognition), and if it does, we identify the source and the target concepts of the metaphor (which we call metaphor explanation). Metaphor explanation is important for understanding metaphors. Explaining type 2 and 3 metaphors is particularly challenging, and, to the best of our knowledge, has not been attempted for nominal concepts¹ before. In our examples, knowing that *life* and *hands* are the target concepts avoids the confusion that may arise if source concepts *sun* and *claws* are used literally in understanding the sentences. This, however, does not mean that the source

¹Nominal concepts are those represented by noun phrases.

concept is a useless embellishment. In the 3rd sentence, knowing that *words* is mapped to *knife* enables the system to understand the emotion or the sentiment embedded in the text. This is the reason why metaphor recognition and explanation is important to applications such as affection mining (Smith et al., 2007).

It is worth noting that some prefer to consider the verb “cut”, rather than the noun “words”, to be metaphoric in the 3rd sentence above. We instead concentrate on nominal metaphors and seek to explain source-target mappings in which at least one domain is a nominal concept. This is because verbs usually have nominal arguments, as either subject or object, thus explaining the source-target mapping of the nominal argument covers most, if not all, cases where a verb is metaphoric.

In order for machines to recognize and explain metaphors, it must have extensive human knowledge. It is not difficult to see why metaphor recognition based on simple context modeling (e.g., by selectional restriction/preference (Resnik, 1993)) is insufficient. First, not all expressions that violate the restriction are metaphors. For example, *I hate to read Heidegger* violates selectional restriction, as the context (embodied by the verb *read*) prefers an object other than a person (*Heidegger*). But, *Heidegger* is not a metaphor but a metonymy, which in this case denotes *Heidegger’s books*. Second, not every metaphor violates the restriction. For example, *life is a journey* is clearly a metaphor, but selectional restriction or preference is helpless when it comes to the isA context.

Existing approaches based on human-curated knowledge bases fall short of the challenge. First, the scale of a human-curated knowledge base is often very limited, which means at best it covers a small set of metaphors. Second, new metaphors are created all the time and the challenge is to recognize and understand metaphors that have never been seen before. This requires extensive knowledge. As a very simple example, even if the machine knows *Sports cars are fire engines* is a metaphor, it still needs to know what is a sports car before it can understand *My Ferrari is a fire engine* is also a metaphor. Third, existing human-curated knowledge bases (including metaphor databases and the WordNet) are not probabilistic. They cannot tell

how typical an instance is of a category (e.g., a *robin* is a more typical bird than a *penguin*), or how popular an expression (e.g., *a breath of fresh air*) is used as a source concept to describe targets in another concept (e.g., *young girls*). Unfortunately, without necessary probabilistic information, not much reasoning can be performed for metaphor explanation.

In this paper, we address the above challenges. We start with a probabilistic isA knowledge base of many entities and categories harnessed from billions of web documents using a set of strict syntactic patterns known as the Hearst patterns (Hearst, 1992). We then automatically acquire a large probabilistic metaphor database with the help of both syntactic patterns and the isA knowledge base (Section 3). Finally we combine the two knowledge bases and a probabilistic reasoning mechanism for automatic metaphor recognition and explanation (Section 4).

This paper makes the following contributions:

1. To our knowledge, we are the first to introduce the metaphor explanation problem, which seeks to recover missing or implied source or target concepts in an implicit metaphor.
2. This is the first big-data driven, unsupervised approach for metaphor recognition and explanation. One of the benefits of leveraging big data is that the knowledge we obtain is less biased, has great coverage, and can be updated in a timely manner. More importantly, a data driven approach can associate with each piece of knowledge probabilities which are not available in human curated knowledge but are indispensable for inference and reasoning.
3. Our results show the effectiveness both in terms of coverage and accuracy of our approach. We manage to acquire one of the largest metaphor knowledge bases ever existed with a precision of 82%. The metaphor recognition accuracy significantly outperforms the state-of-the-art methods (Section 5).

2 Related Work

Existing work on metaphor recognition and interpretation can be divided into two categories: *context-oriented* and *knowledge-driven*. The approach proposed in this paper touches on both categories.

2.1 Context-oriented Methods

Some previous work relies on *context* to differentiate metaphorical expressions from literal ones (Wilks, 1978; Resnik, 1993). The selection restriction theory (Wilks, 1978) argues that the meaning of an expression is restricted by its context, and violations of the restriction imply a metaphor.

Resnik (1993) uses KL divergence to measure the *selectional preference strength (SPS)*, i.e., how strongly a context restricts an expression. Although he did not use this measure directly for metaphor recognition, SPS (and also a related measure called the selection association) is widely used in more recent approaches for metaphor recognition and interpretation (Mason, 2004; Shutova, 2010; Shutova et al., 2010; Baumer et al., 2010). For example, Mason (2004) learns domain-specific selectional preferences and use them to find mappings between concepts from different domains. Shutova (2010) defines metaphor interpretation as a paraphrasing task. The method discriminates between literal and figurative paraphrases by detecting selectional preference violation. The result of this work has been compared with our approach in Section 5. Shutova et al. (2010) identify concepts in a source domain of a metaphor by clustering verb phrases and filtering out verbs that have weak selectional preference strength. Baumer (2010) uses semantic role labeling techniques to calculate selectional preference on semantic relations instead of grammatic relations for metaphor recognition.

A less related but also context-based work is analogy interpretation by relation mapping (Turney, 2008). The problem is to generate mapping between source and target domains by computing pair-wise co-occurrences for different contextual patterns.

Our approach uses selectional restriction when enriching the metaphor knowledge base, and adopts context preference when explaining type 2 and 3 metaphors by focusing on the nearby verbs of a potential source or target concept.

2.2 Knowledge-driven Methods

A growing number of works use knowledge bases for metaphor understanding (Martin, 1990; Narayanan, 1997; Barnden et al., 2002; Veale and Hao, 2008). MIDAS (Martin, 1990) checks if a sen-

tence contains an expression that can be explained by a more general metaphor in a human-curated metaphor knowledge base. ATT-Meta (Barnden et al., 2002) performs metaphor reasoning with a human-curated metaphor knowledge base and first order logic, and it focuses on affection detection (Smith et al., 2007; Agerri, 2008; Zhang, 2010). Krishnakumaran and Zhu (2007) use the isA relation in WordNet (Miller, 1995) for metaphor recognition. Gedigian et al. (2006) use FrameNet (Fillmore et al., 2003) and Probank (Kingsbury and Palmer, 2002) to train a maximum entropy classifier for metaphor recognition. TroFi (Birke and Sarkar, 2006) redefines literal and non-literal as two senses of the same verb and provide two senses with seed sentences from human-curated knowledge bases like WordNet, known metaphor and idiom sets. For a given sentence containing target verb, it compares the similarity of the sentence with two seed sets respectively. If the sentence is closer to the non-literal sense set, the verb is recognized as non-literal usage.

While the above work all relies on human curated data sets or manual labeling, Veale and Hao (2008) introduced the notion of talking points which are figurative properties of noun-based concepts. For example, the concept “*Hamas*” has the following talking points: *is_islamic:movement* and *governs:gaza_strip*. They automatically constructed a knowledge base called *Slip Net* from WordNet and Web corpus. Concepts that are connected on the Slip Net can “slip” to one another and are hence considered related in a metaphor. However, straightforward traversal on the Slip Net can become computationally impractical and the authors did not elaborate on the implementation details. In practice, the knowledge acquired in this paper is much larger but our algorithms are computationally more feasible.

3 Obtaining Probabilistic Knowledge

In this section, we describe how to use a large, general-purpose, probabilistic isA knowledge base Γ_H to create a probabilistic metaphor dataset Γ_m . Γ_H contains isA pairs as well as scores associated with each pair. The metaphor dataset Γ_m contains metaphors of the form: (*source*, *target*), and a weight function P_m that maps a metaphor pair to a probabilistic score. The purpose of creating Γ_H is

to help clean and expand Γ_m , and to perform probabilistic inference for metaphor detection.

3.1 IsA Knowledge Γ_H

Γ_H , a general-purpose, probabilistic isA knowledge base, was previously constructed by Wu et al. (2012).² Γ_H contains isA relations in the form of (x, h_x) , a pair of hyponym and hypernym, for example, *(Steve Ballmer, CEO of IT companies)*, and each pair is associated with a set of probabilistic scores. Two of the most important scores are known as *typicality*: $P(x|h_x)$, the typicality of x of category h_x , and $P(h_x|x)$, the typicality of category h_x for instance x , which will be used in metaphor recognition and explanation. Both scores are approximated by frequencies, e.g.,

$$P(x|h_x) = \frac{\# \text{ of } (x, h_x) \text{ in Hearst extraction}}{\# \text{ of } h_x \text{ in Hearst extraction}}$$

In total, Γ_H contains 16 million unique isA relationships, and 2.7 million unique concepts or categories (the h_x 's in (x, h_x) pairs). The importance of big data is obvious. Γ_H contains millions of categories and probabilistic scores for each category which enables inference for metaphor understanding, as we will show next.

3.2 Acquiring Metaphors Γ_m

We acquire an initial set of metaphors Γ_m from similes. A simile is a figure of speech that explicitly compares two different things using words such as “like” and “as”. For example, the sentence *Life is like a journey* is a simile. Without the word “like,” it becomes a metaphor: *Life is a journey*. This property makes simile an attractive first target for metaphor extraction from a large corpus. We use the following syntactic pattern for extraction:

$$\langle \text{target} \rangle \text{ BE/VB like } [\mathbf{a}] \langle \text{source} \rangle \quad (1)$$

where **BE** denotes *is/are/has been/have been*, etc., **VB** denotes verb other than **BE**, and $\langle \text{target} \rangle$ and $\langle \text{source} \rangle$ denote noun phrases or verb phrases.

Note that not every extracted pair is a metaphor. *Poetry is like an art* matches the pattern, but it is not a metaphor because poetry is really an art. We will use Γ_H to clean such pairs. Furthermore, due to the

²Dataset can be found at <http://probase.msra.cn/>.

idiosyncrasies of natural languages, it is not trivial to correctly extract the $\langle \text{target} \rangle$ and the $\langle \text{source} \rangle$ from each sentence that matches the pattern. We use a postagger and a lemmatizer on the sentences, and we develop a rule-based system that contains more than two dozen rules for extraction. For example, a rule of high-precision but low-recall is “ $\langle \text{target} \rangle$ must be at the beginning of a sentence or the beginning of a clause (e.g., following the word *that*)”.

Finally, from 8,552,672 sentences that match the above pattern (pattern 1), we obtain 1.2 million unique (x, y) pairs, and after filtering, we are left with close to 1 million unique metaphor pairs, which form the starting point of Γ_m .

3.3 Cleaning, Expanding, and Weighting Γ_m

The simile pattern only allows us to extract some of the available metaphor pairs. To expand Γ_m , we use a more flexible but also noisier pattern to extract more candidate metaphor pairs from billions of sentences in the web corpus:

$$\langle \text{target} \rangle \text{ BE } [\mathbf{a}] \langle \text{source} \rangle \quad (2)$$

The above “is a” pattern covers metaphors such as *Life is a journey*. But many pairs thus extracted are not metaphors, for example, *Malaysia is a tropical country*. That is, pairs extracted by the “is a” pattern contains at least two types of relations: the literal isA relations and the metaphor relations. The problem is how to distinguish one from the other. In theory, the set of all IsA relations, I , and the set of all metaphor relations, M , do not overlap, because by definition, the source concept and the target concept in a metaphor are *not* the same thing. Thus, our intuition is the following. The pairs produced by the simile pattern, called S , is a subset of M , while the pairs extracted from the Hearst pattern, called H , is also a subset of I . Since M and I hardly overlap, S and H should have little overlap, too. In practice, very few people would say something like *journeys such as life*. Figure 1 illustrates this scenario.

To verify this intuition, we randomly sampled 1,000 sentences and manually annotated them. Of these sentences, 40 contain an IsA relation, of which 27 are enclosed in a Hearst’s pattern and 13 can be extracted by the “is a” pattern. Furthermore, 28 of these 1000 sentences contain a metaphor expression,

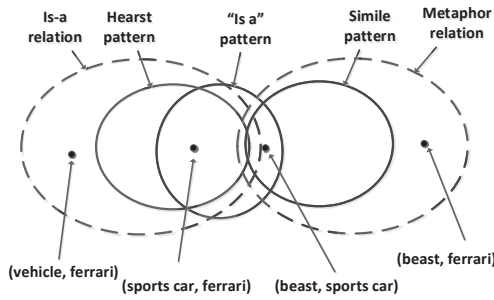


Figure 1: Relations among different sets. Dotted circles represent relations (ground truth). Solid circles represent pairs extracted by syntactic patterns.

and within the 28 metaphors, 15 are embedded in a simile pattern. More importantly, there is no overlap between the IsA relations and metaphors (and hence the similes).

In a larger scale experiment, we crawled 1 billion sentences which match the “is a” pattern (2) from the web corpus. From these, we extracted 180 million unique (x, y) pairs. 24.8% of Γ_H can be found in “is a” pattern pairs, while 16.8% of Γ_m can be found in “is a” pattern pairs. Further more, there is almost no overlap between Γ_H and Γ_m : 1.26% of Γ_H can be found in Γ_m , and 1.31% of Γ_m can be found in Γ_H .

Our goal is to use the information collected through the syntactic patterns to enrich the metaphor relations or Γ_m . Armed with the above observations, we make two conclusions. First, the $(life, journey)$ pair we extracted from *life is a journey* is more likely a metaphor since it does not appear in the set extracted from Hearst patterns. Second, if any existing pair in Γ_m also appears in Γ_H , we can remove that pair from Γ_m .

From the 180 million unique (x, y) pairs we extracted earlier, by filtering out low frequency pairs³ and those pairs in Γ_H , we obtain 2.6 million of fresh metaphors. This is almost 3 times larger than initial metaphor set obtained from the simile pattern.

We further expand Γ_m by adding metaphors derived from Γ_m and Γ_H . Assume $(x, y) \in \Gamma_m$, and $(x, h_x) \in \Gamma_H$, then we add (h_x, y) to Γ_m . As an example, if $(Julie, sun) \in \Gamma_m$,

³Specifically, we randomly sample pairs of frequency 1, 2, ..., 10 from Γ_m and check the precisions of each group. We filter out pairs with frequency less than 5 to optimize the precision.

then we add $(person_name, sun)$ to Γ_m , since $(Julie, person_name) \in \Gamma_H$. This enables the metaphor detection approach we describe in Section 4. Note that we ignore transitivity in the isa relations from Γ_H as such transitivity is not always reliable. For example, car seat is a chair, and chair is furniture, but car seat is not furniture. How to handle transitivity in a data driven isA taxonomy is a challenging problem, and is beyond the scope here.

Finally, we calculate the weight of each metaphor (x, y) . The weight $P_m(x, y)$ is calculated as follows:

$$P_m(x, y) = \frac{\text{occurrences of } (x, y) \text{ in isA pattern}}{\text{occurrences of isA pattern}} \quad (3)$$

The weights of derived metaphors, such as $(person_name, sun)$, are calculated as follows:

$$P_m(h_x, y) = \sum_{(x, h_x) \in \Gamma_H} P_m(x, y) \quad (4)$$

4 Probabilistic Metaphor Understanding

In this paper, we consider two aspects of metaphor understanding, metaphor recognition and metaphor explanation. The latter is needed for type 2 and 3 metaphors where either the source or the target concept is implicit or missing. Next, we describe a probabilistic approach to accomplish these two tasks.

4.1 Type 1 Metaphors

In a type 1 metaphor, both the source and the target concepts appear explicitly. When a sentence matches “is a” pattern (pattern 2), it is a potential metaphor expression. The first noun in the pattern is the target candidate, while the second noun is the source candidate. To recognize type 1 metaphors, we first obtain the candidate (source, target) pair from the sentence. Then, we check if we have any knowledge about the (source, target) pair.

Intuitively, if the pair exists in the metaphor dataset Γ_m , then it is a metaphor. If the pair exists in the is-A knowledge base Γ_H , then it is not a metaphor. But because Γ_m is far from being complete, if a pair exists in neither Γ_m nor Γ_H , there is a possibility that it is a metaphor we have never seen before. In this case, we reason as follows.

Consider a sentence such as *My Ferrari is a beast*. Assume $(Ferrari, beast) \notin \Gamma_m$, but $(sports car,$

beast) $\in \Gamma_m$. Note that (*sports car, beast*) may itself be a derived metaphor which is added into Γ_m in metaphor expansion, and the original metaphor extracted from the web data is (*Lamborghini, beast*). Furthermore, from Γ_H , we know *Ferrari* is a *sports car*, that is, (*Ferrari, sports car*) $\in \Gamma_H$, we can then infer that *Ferrari* to *beast* is very likely a metaphor mapping.

Specifically, let (x, y) be a pair we are concerned with. We want to compute the odds of (x, y) representing a metaphor vs. a normal is-A relationship:

$$\frac{P(x, y)}{1 - P(x, y)} \quad (5)$$

where $P(x, y)$ is the probability that (x, y) forms a metaphor. Now, combining the knowledge we have in Γ_H , we have

$$P(x, y) = \sum_{(x, h_x) \in \Gamma_H} P(x, h_x, y) \quad (6)$$

Here, h_x is a possible superconcept, i.e., a possible interpretation, for x . For example, if $x = \textit{apple}$, then two highly possible interpretations are *company* and *fruit*. In Eq.(6), we want to aggregate on all possible interpretations (all superconcepts) of x . This is possible because of the massive size of the concept space in Γ_H .

We can rewrite Eq.(6) to the following:

$$P(x, y) = \sum_{(x, h_x) \in \Gamma_H} P(y|x, h_x)P(x|h_x)P(h_x) \quad (7)$$

Here, $P(y|x, h_x)$ means when x is interpreted as an h_x , the probability of y as a target metaphorical concept for h_x . Thus, given h_x , y is independent with x , so $P(y|x, h_x)$ can be simply replaced by $P(y|h_x)$. We can then rewrite Eq.(7) to:

$$\begin{aligned} P(x, y) &= \sum_{(x, h_x) \in \Gamma_H} P(y|h_x)P(x|h_x)P(h_x) \\ &= \sum_{(x, h_x) \in \Gamma_H} P(h_x, y)P(x|h_x) \end{aligned} \quad (8)$$

It is clear $P(h_x, y)$ is simply $P_m(h_x, y)$ in Eq.(4) given by the metaphor dataset Γ_m . Furthermore, $P(x|h_x)$ is the typicality of x in the h_x category, and $P(h_x)$ is the prior of the category h_x . Both of them are available from the isA knowledge base Γ_H .

Thus, we can calculate Eq.(8) using information in the two knowledge bases we have created.

If the odds in Eq.(5) is greater than a threshold δ , which is determined empirically to be $\delta = \frac{P(\textit{metaphor})}{P(\textit{isa})}$ ⁴, we declare (x, y) as a metaphor.

4.2 Context Preference Modeling

It is more difficult to recognize metaphors when the source concept or the target concept is not explicitly given in a sentence. In this case, we rely on the context in the sentence.

Given a sentence, we find metaphor candidates and the context. Here, *candidates* are noun phrases in the sentence which can potentially be the target or the source concept of a metaphor, while *context* denotes words that have a grammatic dependency with the candidate. The dependency can be subject-predicate, predicate-object, or modifier-header, etc. The context can be a verb, a noun phrase, or an adjective which has certain preference over the target or source candidate. For example, the word *horse* prefers verbs such as *jump, drink* and *eat*; the word *flower* prefers modifiers such as *red, yellow* and *beautiful*.

In this work, we focus on analyzing the preferences of verbs using subject-predicate or predicate-object relation between the verb and the noun phrases. We select 2,226 most frequent verbs from the web corpus. For each verb, we construct the distribution of noun phrases depend on the verb in the sentences sampled from the web corpus. The noun phrases are restricted to be those that occur in Γ_H .

More specifically, for any noun phrase y that appears in Γ_H , we calculate the following

$$P_r(C|y) = \frac{f_r(y, C)}{\sum_C f_r(y, C)} \quad (9)$$

where $f_r(y, C)$ means the frequency of y and context C with relation r . Note we can build preference distribution for context other than verbs since, in theory, r can be any relation (e.g. modifier-head relation).

4.3 Type 2 and Type 3 Metaphors

If a sentence contains type 2 and type 3 metaphors, either the source or the target concepts in the sen-

⁴This is the ratio between the number of metaphors and is-a pairs in a random sample of "is a" pattern sentences.

tence is missing. For each noun phrase x and a context C in such a sentence, we want to know whether x is of literal or metaphoric use. It is a metaphoric use if the selectional preference of some y , which is a source or target concept of x in Γ_m , is larger than the selectional preference of any super-concept of x in Γ_H , by a factor δ . Formally, there exists a y where $(x, y) \in \Gamma_m$ or $(y, x) \in \Gamma_m$, such that

$$\frac{P(y|x, C)}{P(h|x, C)} \geq \delta, \quad \forall (x, h) \in \Gamma_H. \quad (10)$$

To compute (10), we have

$$\begin{aligned} P(y|x, C) &= \frac{P(x, y, C)}{P(x, C)} \\ &= \frac{P(x, y)P(C|x, y)}{P(x, C)} \end{aligned} \quad (11)$$

Assuming x is a target concept and y is a source concept (a Type 3 metaphor), we can obtain $P(x, y)$ by Eq.(8).⁵ Furthermore, C is independent of x in a type 2 or 3 metaphor, since a metaphor is an unusual use of x (the target) within a given context. Therefore $P(C|x, y) = P(C|y)$, where $P(C|y)$ is available from Eq. (9).

Similarly, we have

$$P(h|x, C) = \frac{P(x, h)P(C|h)}{P(x, C)} \quad (12)$$

where $P(x, h)$ is obtained from Γ_H and $P(C|h)$ is from the context preference distribution.

To explain the metaphor, or uncover the missing concept,

$$\begin{aligned} y^* &= \arg \max_{y \wedge (y, x) \in \Gamma_m} P(y|x, C) \\ &= \arg \max_{y \wedge (y, x) \in \Gamma_m} P(y, x)P(C|y) \end{aligned}$$

As a concrete example, consider sentence *My car drinks gasoline*. There are two possible targets: *car* and *gasoline*. The context for both targets is the verb *drink*. Let $x = \text{car}$. By Eq.(11), we first find all y 's for which $(\text{car}, y) \in \Gamma_m$ or $(y, \text{car}) \in \Gamma_m$. We get terms such as *woman*, *friend*, *gun*, *horse*, etc. When we calculate $P(\text{car}, y)$ by Eq.(8), we also need to find hypernyms of *car* in Γ_H , which

⁵Type 2 metaphors can be handled similarly.

may include *vehicle*, *product*, *asset*, etc. For each candidate y , $P(y|\text{car}, C)$ is calculated by metaphor knowledge $P(x, y)$ and context preference $P(C|y_i)$. Table 1 shows the result. Since the selectional preference of *horse* (from Γ_m) is much larger than other literal uses of *car*, this sentence is recognized as a metaphor, and the missing source concept is *horse*.

Table 1: Log probabilities (M: Metaphor, L:Literal).

Type	y_i	$\log P(y_i, \text{car})$	$\log P(C y_i)$	$\log P(y_i \text{car}, C)$
L	vehicle	-6.2	$-\infty$	$-\infty$
L	product	-6.9	$-\infty$	$-\infty$
L	asset	-6.3	$-\infty$	$-\infty$
M	woman	-8.5	-2.8	-11.3
M	friend	-8.0	-3.0	-11.0
M	gun	-8.4	$-\infty$	$-\infty$
M	horse	-8.2	-2.4	-10.6
...

5 Experimental Result

We evaluate the performance of metaphor acquisition, recognition and explanation in our system and compare it with several state-of-the-art mechanisms.

5.1 Metaphor Acquisition

From the web corpus, we collected 8,552,672 sentences matching the “is like a” pattern (pattern 1) and we extracted 932,621 unique high quality simile mappings from them. These simile mappings became the core of Γ_m . Γ_H contains 16,736,068 unique isA pairs. We also collected 1,131,805,382 sentences matching the “is a” pattern (pattern 2), from which 180,446,190 unique mappings were extracted. These mappings contain both metaphors and isA relations. From there, we identified 2,663,127 pairs of metaphors unseen in the simile set. These new metaphor pairs were added to Γ_m . Random samples show that the precisions of the core metaphor dataset and the whole dataset are 93.5% and 82%, respectively. All of the above datasets, a sample of context preference, as well as the test sets mentioned in this section can be found at <http://adapt.seiee.sjtu.edu.cn/~kzhu/metaphor>.

5.2 Type 1 Metaphor Recognition

We compare our type 1 metaphor recognition with the method (known as KZ) by Krishnakumaran and Zhu (2007). For sentences containing “ x is a y ” pattern, KZ used WordNet to detect whether y is a hypernym of x . If not, then this sentence is considered a metaphor. Our test set is 200 random sentences that match the “ x BE a y ” pattern. We label a sentence in the set as a metaphor if the two nouns connected by BE do not actually have isA relation; or if they do have isA relation but the sentence expressed a strong emotion ⁶.

Table 2: Type 1 metaphor recognition

	Precision	Recall	F1
KZ	13%	30%	18%
Our Approach	73%	66%	69%

The result is summarized in Table 2. KZ does not perform as well due to the small coverage of WordNet taxonomy. Only 33 out of 200 sentences contain a concept x that exists in WordNet and has at least one hypernym. And among these, only 2 sentences contain a y which is the hypernym ancestor of x in WordNet. Clearly, the bottleneck is the scale of WordNet.

5.3 Type 2/3 Metaphor Recognition

For type 2/3 metaphor recognition, we compare our results with three other methods. The first competing method (called SA) employs the *selectional association* proposed by Resnik (1993). Selectional association measures the strength of the connection between a predicate (c) and a term (e) by:

$$A(c, e) = \frac{Pr(e|c) \log \frac{Pr(e|c)}{Pr(e)}}{S(c)}, \quad (13)$$

where

$$\begin{aligned} S(c) &= KL(Pr(e|c)||Pr(e)) \\ &= \sum_e Pr(e|c) \log \frac{Pr(e|c)}{Pr(e)} \end{aligned}$$

Given an NP-predicate pair, if its SA score is less than a threshold α (set to 10^{-4} by empirics), then the pair is recognized as a metaphor context.

⁶For example, “*this man is an animal!*”.

Second competing method (called CP) is the *contextual preference* approach (Resnik, 1993) introduced in Section 4.2. To establish context preference distributions, we randomly select 100 million sentences from the web corpus, parse each sentence using Stanford parser (Group, 2013) to obtain all subject-predicate-object triples, and aggregate the triples to get 33,236,292 subject-predicate pairs and 38,890,877 predicate-object pairs. The occurrences of these pairs are used as context preference. Given a pair of NP-predicate pair, if its context preference score is less than a threshold β (set to 10^{-5} by empirics ⁷), then the pair is considered as metaphoric.

The third competing method (called VH) is a variant of our own algorithm with Γ_m replaced by a metaphor database derived from the Slip Net proposed by Veale and Hao (2008), which we call Γ_{VH} . We built a Slip Net containing 21,451 concept nodes associated with 27,533 distinct talking points. We consider two concepts to be metaphoric if they are at most 5 hops apart on the Slip Net. The choice of 5 hops is a trade-off between precision and recall for Slip Net. We thus created Γ_{VH} with 5,633,760 pairs of concepts.

We sampled 1,000 sentences from the BNC dataset (Clear, 1993) as follows. We prepare a list of 2,945 frequent verbs (and their different forms). For each verb, we obtain at most 5 sentences from BNC dataset which contain this verb as a predicate. At this point, we obtain a total of 22,601 sentences and randomly sample 1,000 sentences to form a test set. Each sentence in the set is then manually labeled as being “metaphor” or “non-metaphor”. We label them according to this procedure:

1. for each verb, we collect the intended use, i.e., the categories of its arguments (subject or object) according to Marriam Webster’s dictionary;
2. if the argument of the verb in the sentence belongs to the intended category, the sentence is labeled “non-metaphor”;
3. if the argument and the intended meaning form a metonymy which uses a part or an attribute to

⁷The authors didn’t specify the choice of α and β , and we pick values which optimize the performance of their algorithms.

represent the whole object, the pair is labeled as “non-metaphor”;

4. else the sentence is labeled as “metaphor”.

Table 3: Type 2/3 metaphor recognition

	Precision	Recall	F1
SA	23%	20%	21%
CP	50%	20%	26%
VH	11%	86%	20%
Our Approach	65%	52%	58%

The results for type 2 and 3 metaphor recognition are shown in Table 3. Our knowledge-based approach significantly outperforms the other peers by F-1 measure. Although VH achieves a good recall, its precision is poor. This is because i) Slip Net construction makes heavy use of sibling terms on the WordNet but sibling terms are not necessarily similar terms; ii) many pairs generated by slipping over the Slip Net are in theory related but are not commonly uttered due to the lack of practical context.

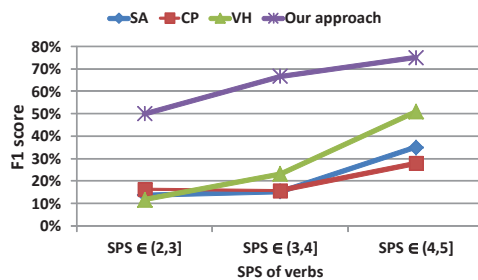


Figure 2: Metaphor recognition of type 2 and 3

Fig. 2 compares the four methods on verbs with different *selectional preference strength*, which indicates how strong a verb’s arguments are restricted to a certain scope of nouns.⁸ Again, our method shows a significant advantage across the board.

We explain why our approach works better using the examples in Table 4. In sentence *AAU200*, *shatters* is a metaphoric usage because silence is not a thing that can be broken into pieces. SA and CP scores for *shatters-silence* pair are high because this word combination is quite common,

⁸Note that no verb has SPS larger than 5.

and hence these methods incorrectly treat it as literal expression. The situation is similar with *stalk-company* pair in *ABG2327*. On the other hand, for *AN81309*, *manipulate-life* is considered rare combination and hence has low SA and CP scores and is deemed a metaphor while in reality it is a literal use. A similar case occurs for *work-concur* pair. In all these cases, our knowledge bases Γ_m and Γ_H are comprehensive and accurate enough to correctly identify metaphors vs. non-metaphors. On the contrary, the metaphor database Γ_{VH} covers way too many pairs that it treats every pair as a metaphor.

Besides our own dataset, we also experiment on TroFi Example Base⁹, which consists of 50 verbs and 3,736 sentences containing these verbs. Each sentence is annotated as literal and nonliteral use of the verb. Our algorithm is used to classify the subjects and the objects of the verbs. We use Stanford dependency parser to obtain collapsed typed dependencies of these sentences, and for each sentence, run our algorithm to classify the subjects and objects related to the verb, if the verb acts as a predicate. Results show that our approach achieves 77.5% precision but just under 5% in recall. The low recall is because, i) non-literal uses in the TroFi dataset include not only metaphor but also metonymy, irony and other anomalies; ii) our approach currently focuses on subject-predicate and predicate-object dependencies in a sentence only, but the target verbs do not act as predicate in many of the example sentences; iii) the Stanford dependency parser is not robust enough so half of the sentences are not parsed correctly.

5.4 Metaphor Explanation

In this experiment, we use the classic labeled metaphoric sentences from (Lakoff and Johnson, 1980). Lakoff provided 24 metaphoric mappings, and for each mapping there are about ten example sentences. In total, there are 214 metaphoric sentences. Among them, we focus on 83 sentences whose metaphor is expressed by subject-predicate or predicate-object relation, as this paper focuses on verb centric context preferences.

We evaluate the results of competing algorithms

⁹TroFi Example Base is available at <http://www.cs.sfu.ca/~anoop/students/jbirke/>.

Table 4: Metaphor recognition for some example sentences from BNC dataset (HM: Human, M: Metaphor, L : Literal).

ID	Sentence	HM	SA	CP	VH	Ours
AAU 200	Road-block salvo <i>shatters</i> Bucharest’s fragile silence .	M	L	L	M	M
ABG 2327	Obstruction and protectionism do not <i>stalk</i> only big companies .	M	L	L	M	M
AN8 1309	But when science proposes to <i>manipulate</i> the life of a human baby,	L	M	M	M	L
ACH 1075	Nevertheless, recent work on Mosley and the BUF has <i>concurred</i> about their basic unimportance.	L	M	M	M	L

by the following labeling criteria. We consider an output (i.e. a pair of concept mapping) as a *match*, if the produced pair exactly matches the ground truth pair, or if the pair is subsumed by the ground truth pair. For example, the ground truth for the sentence *Let that idea simmer on the back burner* is *ideas* \rightarrow *foods* according to Lakoff (Lakoff and Johnson, 1980). If our algorithm outputs *idea* \rightarrow *stew*, then it is considered a *match* since *stew* belongs to the *food* category. An output pair is considered *correct* if it is not a *match* to the ground truth but is otherwise considered metaphoric by at least 2 of the 3 human judges.

Given a sentence, since our algorithm returns a list of possible explanations for the missing concept, ranked by the probability, we evaluate the results by three different metrics:

Match Top 1: result considered correct if there is a *match* with the top explanation;

Match Top 3: result considered correct if there is a *match* in the top 3 ranked explanations;

Correct Top 3: result considered correct if there is a *correct* in the top 3 explanations.

Table 5: Precision of metaphor explanation using different metaphor databases

	Match Top 1	Match Top 3	Correct Top 3
Γ_{VH}	26%	49%	54%
Γ_m	43%	67%	78%

Comparison with Slip Net

We compare the result of our algorithm (from Section 4.3) against the variant which uses Γ_{VH} obtained in Section 5.3.

Table 5 summarizes the precisions of the two algorithms under three different metrics. Some of these sentences and the top explanations given by our algorithm are listed in Table 6. The concept to be explained is italicized while the explanation that is a match or correct is bolded or bold-italicized, respectively. The explanations are ordered from left to right by the score.

Comparison with paraphrasing

While we define metaphor explanation as a task to recover the missing noun-based concept in a source-target mapping, an alternative way to explain a metaphor (Shutova, 2010) is to find the paraphrase of the verb in the metaphor. Here we evaluate paraphrasing task on verbs in metaphoric sentence by Shutova et al (Shutova, 2010). For a metaphoric verb V in a sentence, Shutova et al. select a set of verbs that probabilistically best matches grammar relations of V , and then filter out those verbs that are not related to V according to the WordNet, and eventually re-rank remaining verbs based on selection association.

In some sense, Shutova’s work uses a similar framework as ours: first restrict the target paraphrasing set using a knowledge, then select the most proper word based on the context. The difference is that the target of (Shutova, 2010) is the verb in sentence, while our approach focuses on the noun.

To implement algorithm by Shutova, we extract and count each grammar relation in 1 billion sentences. These counts are used to calculate context matching in (Shutova, 2010), and are also used to calculate selection association. We perform Shutova’s paraphrasing on verbs in 83 sentences, of which only 25 finds a good paraphrases in Shutova’s top 3 results. After removing 17 sentences which contain light verbs (e.g., take, give, put), the algo-

Table 6: Metaphor sentences explained by the system

Metaphor mapping	Sentence	Explanation
Ideas are food	Let that <i>idea</i> simmer on the back burner.	stew; carrot; onion
	We don't need to spoon-feed our students <i>with knowledge</i> .	egg roll; acorn; word
Eyes are containers	His <i>eyes</i> displayed his compassion.	window; symbol; tiny camera
	His <i>eyes</i> were filled with anger.	hollow ball; water balloon; balloon
Emotional effect is physical contact	His <i>mother's death</i> hit him hard.	enemy; monster
	That <i>idea</i> bowled me over.	punch; stew; onion
Life is a container.	Her <i>life</i> is crammed with activities.	tapestry; beach; dance
	Get the most out of <i>life</i> .	game; journey; prison

rithm finds 21 good paraphrases in top 3 results. One reason for the low recall is that Wordnet is inadequate in providing candidate metaphor mapping. This is also the reason why our metaphor base is better than the metaphor base generated by talking points.

6 Conclusion

Knowledge is essential for a machine to identify and understand metaphors. In this paper, we show how to make use of two probabilistic knowledge bases automatically acquired from billions of web pages for this purpose. This work currently recognizes and explains metaphoric mappings between nominal concepts with the help of selectional preference of just subject-predicate or predicate-object contexts. An immediate next step is to extend this framework to more general contexts and a further improvement will be to identify mappings between any source and target domains.

7 Acknowledgements

Kenny Q. Zhu was partially supported by Google Faculty Research Award, and NSFC Grants 61100050, 61033002 and 61373031.

References

- Rodrigo Agerri. 2008. Metaphor in textual entailment. In *COLING (Posters)*, pages 3–6.
- John Barnden, Sheila Glasbey, Mark Lee, and Alan Wallington. 2002. Reasoning in metaphor understanding: the att-meta approach and system. In *COLING '02*, pages 1–5.
- Eric P. S. Baumer, James P. White, and Bill Tomlinson. 2010. Comparing semantic role labeling with typed dependency parsing in computational metaphor identification. In *CALC '10*, pages 14–22.
- Julia Birke and Anoop Sarkar. 2006. A clustering approach for nearly unsupervised recognition of nonliteral language. In *In Proceedings of EACL-06*, pages 329–336.
- Jeremy H. Clear. 1993. The digital word. chapter The British national corpus, pages 163–187.
- Charles J. Fillmore, Christopher R. Johnson, and Miriam R.L. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography*, 16.3:235–250.
- James Geary. 2011. *I is an Other: The Secret Life of Metaphor and How It Shapes the Way We See the World*. Harper.
- Matt Gedigian, John Bryant, Srinu Narayanan, and Branimir Cicic. 2006. Catching metaphors. In *In Workshop On Scalable Natural Language Understanding*.
- Stanford NLP Group. 2013. The Stanford parser. <http://nlp.stanford.edu/software/lex-parser.shtml>.
- Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING '92*, pages 539–545.
- Paul Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *In Language Resources and Evaluation*.
- Saisuresh Krishnakumaran and Xiaojin Zhu. 2007. Hunting elusive metaphors using lexical resources.

- In *Proceedings of the Workshop on Computational Approaches to Figurative Language*, pages 13–20, Rochester, New York, April. ACL.
- George Lakoff and Mark Johnson. 1980. *Metaphors We Live By*. University of Chicago Press, Chicago, USA.
- J. H. Martin. 1990. *A Computational Model of Metaphor Interpretation*. Academic Press Professional, Inc.
- Zachary J. Mason. 2004. Cornet: a computational, corpus-based conventional metaphor extraction system. *Comput. Linguist.*, 30:23–44, March.
- George A. Miller. 1995. Wordnet: a lexical database for english. *Commun. ACM*, 38:39–41, November.
- Srinivas Sankara Narayanan. 1997. Knowledge-based action representations for metaphor and aspect (karma). Technical report.
- Philip Stuart Resnik. 1993. *Selection and information: a class-based approach to lexical relationships*. Ph.D. thesis.
- Ekaterina Shutova, Lin Sun, and Anna Korhonen. 2010. Metaphor identification using verb and noun clustering. In *COLING '10*, pages 1002–1010.
- Ekaterina Shutova. 2010. Automatic metaphor interpretation as a paraphrasing task. In *HLT '10*, pages 1029–1037.
- Catherine Smith, Tim Rumbell, John Barnden, Bob Hendley, Mark Lee, and Alan Wallington. 2007. Don't worry about metaphor: affect extraction for conversational agents. In *ACL '07*, pages 37–40.
- P.D. Turney. 2008. The latent relation mapping engine: Algorithm and experiments. *Journal of Artificial Intelligence Research*, 33(1):615–655.
- Tony Veale and Yanfen Hao. 2008. A fluid knowledge representation for understanding and generating creative metaphors. In *COLING*, pages 945–952.
- Yorick Wilks. 1978. Making preferences more active. *Artificial Intelligence*, 11(3):197 – 223.
- Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Qili Zhu. 2012. Probbase: a probabilistic taxonomy for text understanding. In *SIGMOD Conference*, pages 481–492.
- Li Zhang. 2010. Metaphor interpretation and context-based affect detection. In *COLING (Posters)*, pages 1480–1488.

Grounded Compositional Semantics for Finding and Describing Images with Sentences

Richard Socher, Andrej Karpathy, Quoc V. Le*, Christopher D. Manning, Andrew Y. Ng

Stanford University, Computer Science Department, *Google Inc.

richard@socher.org, karpathy@cs.stanford.edu,

qvl@google.com, manning@stanford.edu, ang@cs.stanford.edu

Abstract

Previous work on Recursive Neural Networks (RNNs) shows that these models can produce compositional feature vectors for accurately representing and classifying sentences or images. However, the sentence vectors of previous models cannot accurately represent visually grounded meaning. We introduce the DT-RNN model which uses dependency trees to embed sentences into a vector space in order to retrieve images that are described by those sentences. Unlike previous RNN-based models which use constituency trees, DT-RNNs naturally focus on the action and agents in a sentence. They are better able to abstract from the details of word order and syntactic expression. DT-RNNs outperform other recursive and recurrent neural networks, kernelized CCA and a bag-of-words baseline on the tasks of finding an image that fits a sentence description and vice versa. They also give more similar representations to sentences that describe the same image.

1 Introduction

Single word vector spaces are widely used (Turney and Pantel, 2010) and successful at classifying single words and capturing their meaning (Collobert and Weston, 2008; Huang et al., 2012; Mikolov et al., 2013). Since words rarely appear in isolation, the task of learning compositional meaning representations for longer phrases has recently received a lot of attention (Mitchell and Lapata, 2010; Socher et al., 2010; Socher et al., 2012; Grefenstette et al., 2013). Similarly, classifying whole images into a

fixed set of classes also achieves very high performance (Le et al., 2012; Krizhevsky et al., 2012). However, similar to words, objects in images are often seen in relationships with other objects which are not adequately described by a single label.

In this work, we introduce a model, illustrated in Fig. 1, which learns to map sentences and images into a common embedding space in order to be able to retrieve one from the other. We assume word and image representations are first learned in their respective single modalities but finally mapped into a jointly learned multimodal embedding space.

Our model for mapping sentences into this space is based on ideas from Recursive Neural Networks (RNNs) (Pollack, 1990; Costa et al., 2003; Socher et al., 2011b). However, unlike all previous RNN models which are based on constituency trees (CT-RNNs), our model computes compositional vector representations inside dependency trees. The compositional vectors computed by this new dependency tree RNN (DT-RNN) capture more of the meaning of sentences, where we define meaning in terms of similarity to a “visual representation” of the textual description. DT-RNN induced vector representations of sentences are more robust to changes in the syntactic structure or word order than related models such as CT-RNNs or Recurrent Neural Networks since they naturally focus on a sentence’s action and its agents.

We evaluate and compare DT-RNN induced representations on their ability to use a sentence such as “*A man wearing a helmet jumps on his bike near a beach.*” to find images that show such a scene. The goal is to learn sentence representations that capture

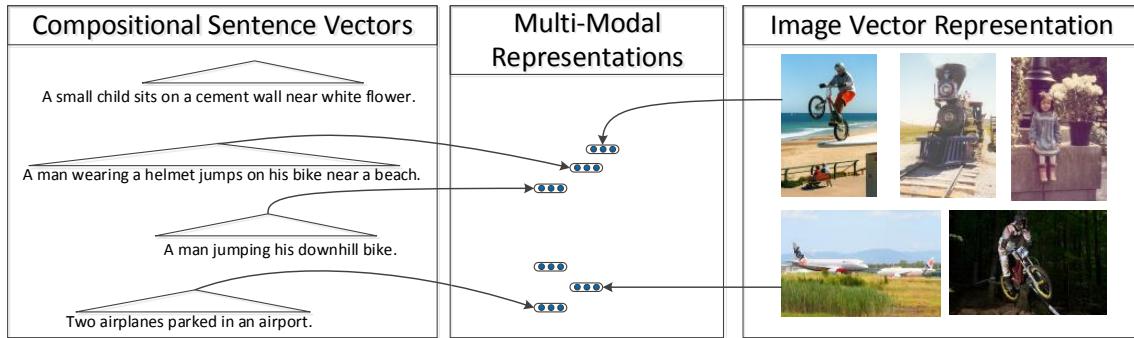


Figure 1: The DT-RNN learns vector representations for sentences based on their dependency trees. We learn to map the outputs of convolutional neural networks applied to images into the same space and can then compare both sentences and images. This allows us to query images with a sentence and give sentence descriptions to images.

the visual scene described and to find appropriate images in the learned, multi-modal sentence-image space. Conversely, when given a query image, we would like to find a description that goes beyond a single label by providing a correct sentence describing it, a task that has recently garnered a lot of attention (Farhadi et al., 2010; Ordonez et al., 2011; Kuznetsova et al., 2012). We use the dataset introduced by (Rashtchian et al., 2010) which consists of 1000 images, each with 5 descriptions. On all tasks, our model outperforms baselines and related models.

2 Related Work

The presented model is connected to several areas of NLP and vision research, each with a large amount of related work to which we can only do some justice given space constraints.

Semantic Vector Spaces and Their Compositionality. The dominant approach in semantic vector spaces uses distributional similarities of single words. Often, co-occurrence statistics of a word and its context are used to describe each word (Turney and Pantel, 2010; Baroni and Lenci, 2010), such as tf-idf. Most of the compositionality algorithms and related datasets capture two-word compositions. For instance, (Mitchell and Lapata, 2010) use two-word phrases and analyze similarities computed by vector addition, multiplication and others. Compositionality is an active field of research with many different models and representations being explored (Grefenstette et al., 2013), among many others. We compare to supervised compositional models that

can learn task-specific vector representations such as constituency tree recursive neural networks (Socher et al., 2011b; Socher et al., 2011a), chain structured recurrent neural networks and other baselines. Another alternative would be to use CCG trees as a backbone for vector composition (K.M. Hermann, 2013).

Multimodal Embeddings. Multimodal embedding methods project data from multiple sources such as sound and video (Ngiam et al., 2011) or images and text. Socher et al. (Socher and Fei-Fei, 2010) project words and image regions into a common space using kernelized canonical correlation analysis to obtain state of the art performance in annotation and segmentation. Similar to our work, they use unsupervised large text corpora to learn semantic word representations. Among other recent work is that by Srivastava and Salakhutdinov (2012) who developed multimodal Deep Boltzmann Machines. Similar to their work, we use techniques from the broad field of deep learning to represent images and words.

Recently, single word vector embeddings have been used for zero shot learning (Socher et al., 2013c). Mapping images to word vectors enabled their system to classify images as depicting objects such as "cat" without seeing any examples of this class. Related work has also been presented at NIPS (Socher et al., 2013b; Frome et al., 2013). This work moves zero-shot learning beyond single categories per image and extends it to unseen phrases and full length sentences, making use of similar ideas of semantic spaces grounded in visual knowledge.

Detailed Image Annotation. Interactions between images and texts is a growing research field. Early work in this area includes generating single words or fixed phrases from images (Duygulu et al., 2002; Barnard et al., 2003) or using contextual information to improve recognition (Gupta and Davis, 2008; Torralba et al., 2010).

Apart from a large body of work on single object image classification (Le et al., 2012), there is also work on attribute classification and other mid-level elements (Kumar et al., 2009), some of which we hope to capture with our approach as well.

Our work is close in spirit with recent work in describing images with more detailed, longer textual descriptions. In particular, Yao et al. (2010) describe images using hierarchical knowledge and humans in the loop. In contrast, our work does not require human interactions. Farhadi et al. (2010) and Kulkarni et al. (2011), on the other hand, use a more automatic method to parse images. For instance, the former approach uses a single triple of objects estimated for an image to retrieve sentences from a collection written to describe similar images. It forms representations to describe 1 object, 1 action, and 1 scene. Kulkarni et al. (2011) extends their method to describe an image with multiple objects. None of these approaches have used a compositional sentence vector representation and they require specific language generation techniques and sophisticated inference methods. Since our model is based on neural networks inference is fast and simple. Kuznetsova et al. (2012) use a very large parallel corpus to connect images and sentences. Feng and Lapata (2013) use a large dataset of captioned images and experiments with both extractive (search) and abstractive (generation) models.

Most related is the very recent work of Hodosh et al. (2013). They too evaluate using a ranking measure. In our experiments, we compare to kernelized Canonical Correlation Analysis which is the main technique in their experiments.

3 Dependency-Tree Recursive Neural Networks

In this section we first focus on the DT-RNN model that computes compositional vector representations for phrases and sentences of variable length and syn-

tactic type. In section 5 the resulting vectors will then become multimodal features by mapping images that show what the sentence describes to the same space and learning both the image and sentence mapping jointly.

The most common way of building representations for longer phrases from single word vectors is to simply linearly average the word vectors. While this bag-of-words approach can yield reasonable performance in some tasks, it gives all the words the same weight and cannot distinguish important differences in simple visual descriptions such as *The bike crashed into the standing car.* vs. *The car crashed into the standing bike.*

RNN models (Pollack, 1990; Goller and Küchler, 1996; Socher et al., 2011b; Socher et al., 2011a) provided a novel way of combining word vectors for longer phrases that moved beyond simple averaging. They combine vectors with an RNN in binary constituency trees which have potentially many hidden layers. While the induced vector representations work very well on many tasks, they also inevitably capture a lot of syntactic structure of the sentence. However, the task of finding images from sentence descriptions requires us to be more invariant to syntactic differences. One such example are active-passive constructions which can collapse words such as “by” in some formalisms (de Marneffe et al., 2006), relying instead on the semantic relationship of “agent”. For instance, *The mother hugged her child.* and *The child was hugged by its mother.* should map to roughly the same visual space. Current Recursive and Recurrent Neural Networks do not exhibit this behavior and even bag of words representations would be influenced by the words *was* and *by*. The model we describe below focuses more on recognizing actions and agents and has the potential to learn representations that are invariant to active-passive differences.

3.1 DT-RNN Inputs: Word Vectors and Dependency Trees

In order for the DT-RNN to compute a vector representation for an ordered list of m words (a phrase or sentence), we map the single words to a vector space and then parse the sentence.

First, we map each word to a d -dimensional vector. We initialize these word vectors with the un-

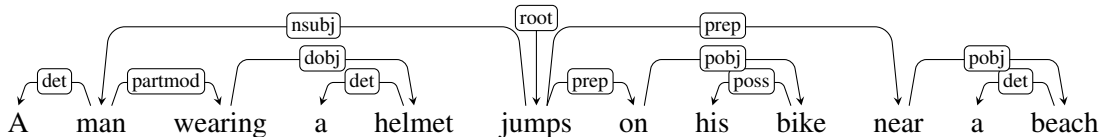


Figure 2: Example of a full dependency tree for a longer sentence. The DT-RNN will compute vector representations at every word that represents that word and an arbitrary number of child nodes. The final representation is computed at the *root* node, here at the verb *jumps*. Note that more important activity and object words are higher up in this tree structure.

supervised model of Huang et al. (2012) which can learn single word vector representations from both local and global contexts. The idea is to construct a neural network that outputs high scores for windows and documents that occur in a large unlabeled corpus and low scores for window-document pairs where one word is replaced by a random word. When such a network is optimized via gradient descent the derivatives backpropagate into a word embedding matrix A which stores word vectors as columns. In order to predict correct scores the vectors in the matrix capture co-occurrence statistics. We use $d = 50$ in all our experiments. The embedding matrix X is then used by finding the column index i of each word: $[w] = i$ and retrieving the corresponding column x_w from X . Henceforth, we represent an input sentence s as an ordered list of (word,vector) pairs: $s = ((w_1, x_{w_1}), \dots, (w_m, x_{w_m}))$.

Next, the sequence of words (w_1, \dots, w_m) is parsed by the dependency parser of de Marneffe et al. (2006). Fig. 2 shows an example. We can represent a dependency tree d of a sentence s as an ordered list of (child,parent) indices: $d(s) = \{(i, j)\}$, where every child word in the sequence $i = 1, \dots, m$ is present and has any word $j \in \{1, \dots, m\} \cup \{0\}$ as its parent. The root word has as its parent 0 and we notice that the same word can be a parent between zero and m number of times. Without loss of generality, we assume that these indices form a tree structure. To summarize, the input to the DT-RNN for each sentence is the pair (s, d) : the words and their vectors and the dependency tree.

3.2 Forward Propagation in DT-RNNs

Given these two inputs, we now illustrate how the DT-RNN computes parent vectors. We will use the following sentence as a running example: *Students₁ ride₂ bikes₃ at₄ night₅*. Fig. 3 shows its tree and computed vector representations. The depen-

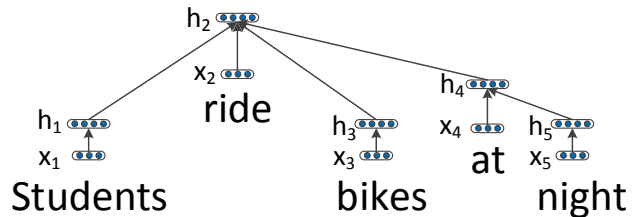


Figure 3: Example of a DT-RNN tree structure for computing a sentence representation in a bottom up fashion.

dependency tree for this sentence can be summarized by the following set of (child, parent) edges: $d = \{(1, 2), (2, 0), (3, 2), (4, 2), (5, 4)\}$.

The DT-RNN model will compute parent vectors at each word that include all the dependent (children) nodes in a bottom up fashion using a compositionality function g_θ which is parameterized by all the model parameters θ . To this end, the algorithm searches for nodes in a tree that have either (i) no children or (ii) whose children have already been computed and then computes the corresponding vector.

In our example, the words x_1, x_3, x_5 are leaf nodes and hence, we can compute their corresponding hidden nodes via:

$$h_c = g_\theta(x_c) = f(W_v x_c) \quad \text{for } c = 1, 3, 5, \quad (1)$$

where we compute the hidden vector at position c via our general composition function g_θ . In the case of leaf nodes, this composition function becomes simply a linear layer, parameterized by $W_v \in \mathbb{R}^{n \times d}$, followed by a nonlinearity. We cross-validate over using no nonlinearity ($f = \text{id}$), tanh, sigmoid or rectified linear units ($f = \max(0, x)$), but generally find tanh to perform best.

The final sentence representation we want to compute is at h_2 , however, since we still do not have h_4 ,

we compute that one next:

$$h_4 = g_\theta(x_4, h_5) = f(W_v x_4 + W_{r1} h_5), \quad (2)$$

where we use the same W_v as before to map the word vector into hidden space but we now also have a linear layer that takes as input h_5 , the only child of the fourth node. The matrix $W_{r1} \in \mathbb{R}^{n \times n}$ is used because node 5 is the first child node on the right side of node 4. Generally, we have multiple matrices for composing with hidden child vectors from the right and left sides: $W_r = (W_{r1}, \dots, W_{rk_r})$ and $W_l = (W_{l1}, \dots, W_{lk_l})$. The number of needed matrices is determined by the data by simply finding the maximum numbers of left k_l and right k_r children any node has. If at test time a child appeared at an even large distance (this does not happen in our test set), the corresponding matrix would be the identity matrix.

Now that all children of h_2 have their hidden vectors, we can compute the final sentence representation via:

$$h_2 = g_\theta(x_2, h_1, h_3, h_4) = f(W_v x_2 + W_{l1} h_1 + W_{r1} h_3 + W_{r2} h_4). \quad (3)$$

Notice that the children are multiplied by matrices that depend on their location relative to the current node.

Another modification that improves the mean rank by approximately 6 in image search on the dev set is to weight nodes by the number of words underneath them and normalize by the sum of words under all children. This encourages the intuitive desideratum that nodes describing longer phrases are more important. Let $\ell(i)$ be the number of leaf nodes (words) under node i and $C(i, y)$ be the set of child nodes of node i in dependency tree y . The final composition function for a node vector h_i becomes:

$$h_i = f \left(\frac{1}{\ell(i)} \left(W_v x_i + \sum_{j \in C(i)} \ell(j) W_{\text{pos}(i,j)} h_j \right) \right), \quad (4)$$

where by definition $\ell(i) = 1 + \sum_{j \in C(i)} \ell(j)$ and $\text{pos}(i, j)$ is the relative position of child j with respect to node i , e.g. $l1$ or $r2$ in Eq. 3.

3.3 Semantic Dependency Tree RNNs

An alternative is to condition the weight matrices on the semantic relations given by the dependency

parser. We use the collapsed tree formalism of the Stanford dependency parser (de Marneffe et al., 2006). With such a semantic untying of the weights, the DT-RNN makes better use of the dependency formalism and could give active-passive reversals similar semantic vector representation. The equation for this semantic DT-RNN (**SDT-RNN**) is the same as the one above except that the matrices $W_{\text{pos}(i,j)}$ are replaced with matrices based on the dependency relationship. There are a total of 141 unique such relationships in the dataset. However, most are very rare. For examples of semantic relationships, see Fig. 2 and the model analysis section 6.7.

This forward propagation can be used for computing compositional vectors and in Sec. 5 we will explain the objective function in which these are trained.

3.4 Comparison to Previous RNN Models

The DT-RNN has several important differences to previous RNN models of Socher et al. (2011a) and (Socher et al., 2011b; Socher et al., 2011c). These constituency tree RNNs (CT-RNNs) use the following composition function to compute a hidden parent vector h from exactly two child vectors (c_1, c_2) in a binary tree: $h = f \left(W \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \right)$, where $W \in \mathbb{R}^{d \times 2d}$ is the main parameter to learn. This can be rewritten to show the similarity to the DT-RNN as $h = f(W_{l1} c_1 + W_{r1} c_2)$. However, there are several important differences.

Note first that in previous RNN models the parent vectors were of the same dimensionality to be recursively compatible and be used as input to the next composition. In contrast, our new model first maps single words into a hidden space and then parent nodes are composed from these hidden vectors. This allows a higher capacity representation which is especially helpful for nodes that have many children.

Secondly, the DT-RNN allows for n -ary nodes in the tree. This is an improvement that is possible even for constituency tree CT-RNNs but it has not been explored in previous models.

Third, due to computing parent nodes in constituency trees, previous models had the problem that words that are merged last in the tree have a larger weight or importance in the final sentence rep-

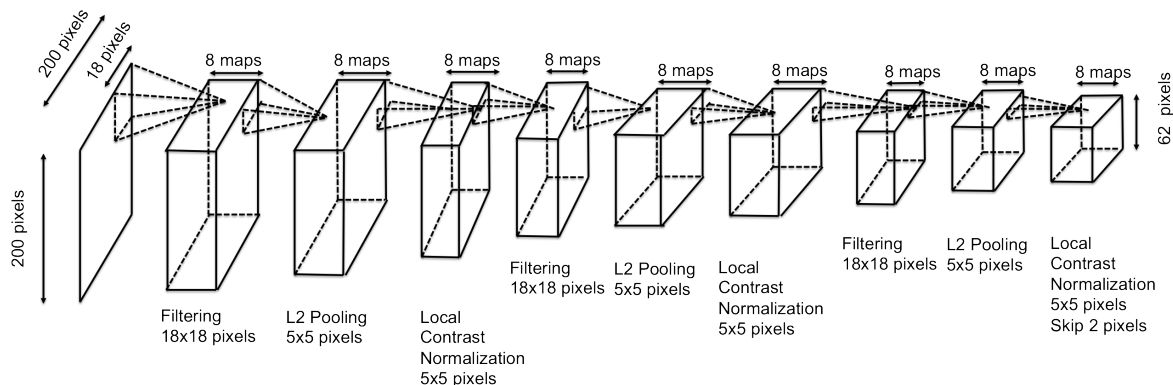


Figure 4: The architecture of the visual model. This model has 3 sequences of filtering, pooling and local contrast normalization layers. The learnable parameters are the filtering layer. The filters are not shared, i.e., the network is nonconvolutional.

resentation. This can be problematic since these are often simple non-content words, such as a leading ‘But,’. While such single words can be important for tasks such as sentiment analysis, we argue that for describing visual scenes the DT-RNN captures the more important effects: The dependency tree structures push the central content words such as the main action or verb and its subject and object to be merged last and hence, by construction, the final sentence representation is more robust to less important adjectival modifiers, word order changes, etc.

Fourth, we allow some untying of weights depending on either how far away a constituent is from the current word or what its semantic relationship is.

Now that we can compute compositional vector representations for sentences, the next section describes how we represent images.

4 Learning Image Representations with Neural Networks

The image features that we use in our experiments are extracted from a deep neural network, replicated from the one described in (Le et al., 2012). The network was trained using both unlabeled data (random web images) and labeled data to classify 22,000 categories in ImageNet (Deng et al., 2009). We then used the features at the last layer, before the classifier, as the feature representation in our experiments. The dimension of the feature vector of the last layer is 4,096. The details of the model and its training procedures are as follows.

The architecture of the network can be seen in Figure 4. The network takes 200x200 pixel images as inputs and has 9 layers. The layers consist of

three sequences of filtering, pooling and local contrast normalization (Jarrett et al., 2009). The pooling function is L2 pooling of the previous layer (taking the square of the filtering units, summing them up in a small area in the image, and taking the square-root). The local contrast normalization takes inputs in a small area of the lower layer, subtracts the mean and divides by the standard deviation.

The network was first trained using an unsupervised objective: trying to reconstruct the input while keeping the neurons sparse. In this phase, the network was trained on 20 million images randomly sampled from the web. We resized a given image so that its short dimension has 200 pixels. We then cropped a fixed size 200x200 pixel image right at the center of the resized image. This means we may discard a fraction of the long dimension of the image.

After unsupervised training, we used ImageNet (Deng et al., 2009) to adjust the features in the entire network. The ImageNet dataset has 22,000 categories and 14 million images. The number of images in each category is equal across categories. The 22,000 categories are extracted from WordNet.

To speed up the supervised training of this network, we made a simple modification to the algorithm described in Le et al. (2012): adding a “bottleneck” layer in between the last layer and the classifier. to reduce the number of connections. We added one “bottleneck” layer which has 4,096 units in between the last layer of the network and the softmax layer. This newly-added layer is fully connected to the previous layer and has a linear activation function. The total number of connections of this network is approximately 1.36 billion.

The network was trained again using the supervised objective of classifying the 22,000 classes in ImageNet. Most features in the networks are local, which allows model parallelism. Data parallelism by asynchronous SGD was also employed as in Le et al. (2012). The entire training, both unsupervised and supervised, took 8 days on a large cluster of machines. This network achieves 18.3% precision@1 on the full ImageNet dataset (Release Fall 2011).

We will use the features at the bottleneck layer as the feature vector z of an image. Each scaled and cropped image is presented to our network. The network then performs a feedforward computation to compute the values of the bottleneck layer. This means that every image is represented by a fixed length vector of 4,096 dimensions. Note that during training, no aligned sentence-image data was used and the ImageNet classes do not fully intersect with the words used in our dataset.

5 Multimodal Mappings

The previous two sections described how we can map sentences into a $d = 50$ -dimensional space and how to extract high quality image feature vectors of 4096 dimensions. We now define our final multimodal objective function for learning joint image-sentence representations with these models. Our training set consists of N images and their feature vectors z_i and each image has 5 sentence descriptions s_{i1}, \dots, s_{i5} for which we use the DT-RNN to compute vector representations. See Fig. 5 for examples from the dataset. For training, we use a max-margin objective function which intuitively trains pairs of correct image and sentence vectors to have high inner products and incorrect pairs to have low inner products. Let $v_i = W_I z_i$ be the mapped image vector and $y_{ij} = DTRNN_{\theta}(s_{ij})$ the composed sentence vector. We define \mathcal{S} to be the set of all sentence indices and $\mathcal{S}(i)$ the set of sentence indices corresponding to image i . Similarly, \mathcal{I} is the set of all image indices and $\mathcal{I}(j)$ is the image index of sentence j . The set \mathcal{P} is the set of all correct image-sentence training pairs (i, j) . The ranking cost function to minimize is then: $J(W_I, \theta) =$

$$\sum_{(i,j) \in \mathcal{P}} \sum_{c \in \mathcal{S} \setminus \mathcal{S}(i)} \max(0, \Delta - v_i^T y_j + v_i^T y_c) + \sum_{(i,j) \in \mathcal{P}} \sum_{c \in \mathcal{I} \setminus \mathcal{I}(j)} \max(0, \Delta - v_i^T y_j + v_c^T y_j), \quad (5)$$

where θ are the language composition matrices, and both second sums are over other sentences coming from different images and vice versa. The hyperparameter Δ is the margin. The margin is found via cross validation on the dev set and usually around 1.

The final objective also includes the regularization term $\lambda/lef t(\|\theta\|_2^2 + \|W_I\|_F)$. Both the visual model and the word vector learning require a very large amount of training data and both have a huge number of parameters. Hence, to prevent overfitting, we assume their weights are fixed and only train the DT-RNN parameters W_I . If larger training corpora become available in the future, training both jointly becomes feasible and would present a very promising direction. We use a modified version of AdaGrad (Duchi et al., 2011) for optimization of both W_I and the DT-RNN as well as the other baselines (except kCCA). Adagrad has achieved good performance previously in neural networks models (Dean et al., 2012; Socher et al., 2013a). We modify it by resetting all squared gradient sums to 1 every 5 epochs. With both images and sentences in the same multimodal space, we can easily query the model for similar images or sentences by finding the nearest neighbors in terms of negative inner products.

An alternative objective function is based on the squared loss $J(W_I, \theta) = \sum_{(i,j) \in \mathcal{P}} \|v_i - y_j\|_2^2$. This requires an alternating minimization scheme that first trains only W_I , then fixes W_I and trains the DT-RNN weights θ and then repeats this several times. We find that the performance with this objective function (paired with finding similar images using Euclidean distances) is worse for all models than the margin loss of Eq. 5. In addition kCCA also performs much better using inner products in the multimodal space.

6 Experiments

We use the dataset of Rashtchian et al. (2010) which consists of 1000 images, each with 5 sentences. See Fig. 5 for examples.

We evaluate and compare the DT-RNN in three different experiments. First, we analyze how well the sentence vectors capture similarity in visual meaning. Then we analyze *Image Search with Query Sentences*: to query each model with a sentence in order to find an image showing that sen-



1. A woman and her dog watch the cameraman in their living with wooden floors.
2. A woman sitting on the couch while a black faced dog runs across the floor.
3. A woman wearing a backpack sits on a couch while a small dog runs on the hardwood floor next to her.
4. A women sitting on a sofa while a small Jack Russell walks towards the camera.
5. White and black small dog walks toward the camera while woman sits on couch, desk and computer seen in the background as well as a pillow, teddy bear and moggie toy on the wood floor.



1. A man in a cowboy hat check approaches a small red sports car.
2. The back and left side of a red Ferrari and two men admiring it.
3. The sporty car is admired by passer by.
4. Two men next to a red sports car in a parking lot.
5. Two men stand beside a red sports car.

Figure 5: Examples from the dataset of images and their sentence descriptions (Rashtchian et al., 2010). Sentence length varies greatly and different objects can be mentioned first. Hence, models have to be invariant to word ordering.

tence’s visual ‘meaning.’ The last experiment *Describing Images by Finding Suitable Sentences* does the reverse search where we query the model with an image and try to find the closest textual description in the embedding space.

In our comparison to other methods we focus on those models that can also compute fixed, continuous vectors for sentences. In particular, we compare to the RNN model on constituency trees of Socher et al. (2011a), a standard recurrent neural network; a simple bag-of-words baseline which averages the words. All models use the word vectors provided by Huang et al. (2012) and do not update them as discussed above. Models are trained with their corresponding gradients and backpropagation techniques. A standard recurrent model is used where the hidden vector at word index t is computed from the hidden vector at the previous time step and the current word vector: $h_t = f(W_h h_{t-1} + W_x x_t)$. During training, we take the last hidden vector of the sentence chain and propagate the error into that. It is also this vector that is used to represent the sentence.

Other possible comparisons are to the very different models mentioned in the related work section. These models use a lot more task-specific engineering, such as running object detectors with bounding boxes, attribute classifiers, scene classifiers, CRFs for composing the sentences, etc. Another line of work uses large sentence-image aligned resources (Kuznetsova et al., 2012), whereas we focus on easily obtainable training data of each modality separately and a rather small multimodal corpus.

In our experiments we split the data into 800 training, 100 development and 100 test images. Since there are 5 sentences describing each image, we

have 4000 training sentences and 500 testing sentences. The dataset has 3020 unique words, half of which only appear once. Hence, the unsupervised, pre-trained semantic word vector representations are crucial. Word vectors are not fine tuned during training. Hence, the main parameters are the DT-RNN’s W_l, W_r , or the semantic matrices of which there are 141 and the image mapping W_I . For both DT-RNNs the weight matrices are initialized to block identity matrices plus Gaussian noise. Word vectors and hidden vectors are set o length 50. Using the development split, we found $\lambda = 0.08$ and the learning rate of AdaGrad to 0.0001. The best model uses a margin of $\Delta = 3$.

Inspired by Socher and Fei-Fei (2010) and Hosh et al. (2013) we also compare to kernelized Canonical Correlation Analysis (kCCA). We use the average of word vectors for describing sentences and the same powerful image vectors as before. We use the code of Socher and Fei-Fei (2010). Technically, one could combine the recently introduced deep CCA Andrew et al. (2013) and train the recursive neural network architectures with the CCA objective. We leave this to future work. With linear kernels, kCCA does well for image search but is worse for sentence self similarity and describing images with sentences close-by in embedding space. All other models are trained by replacing the DT-RNN function in Eq. 5.

6.1 Similarity of Sentences Describing the Same Image

In this experiment, we first map all 500 sentences from the test set into the multi-modal space. Then for each sentence, we find the nearest neighbor sen-

<i>Sentences Similarity for Image</i>		<i>Image Search</i>		<i>Describing Images</i>	
Model	Mean Rank	Model	Mean Rank	Model	Mean Rank
Random	101.1	Random	52.1	Random	92.1
BoW	11.8	BoW	14.6	BoW	21.1
CT-RNN	15.8	CT-RNN	16.1	CT-RNN	23.9
Recurrent NN	18.5	Recurrent NN	19.2	Recurrent NN	27.1
kCCA	10.7	kCCA	15.9	kCCA	18.0
DT-RNN	11.1	DT-RNN	13.6	DT-RNN	19.2
SDT-RNN	10.5	SDT-RNN	12.5	SDT-RNN	16.9

Table 1: **Left:** Comparison of methods for sentence similarity judgments. Lower numbers are better since they indicate that sentences describing the same image rank more highly (are closer). The ranks are out of the 500 sentences in the test set. **Center:** Comparison of methods for image search with query sentences. Shown is the average rank of the single correct image that is being described. **Right:** Average rank of a correct sentence description for a query image.

tences in terms of inner products. We then sort these neighbors and record the rank or position of the nearest sentence *that describes the same image*. If all the images were very unique and the visual descriptions close-paraphrases and consistent, we would expect a very low rank. However, usually a handful of images are quite similar (for instance, there are various images of airplanes flying, parking, taxiing or waiting on the runway) and sentence descriptions can vary greatly in detail and specificity for the same image.

Table 1 (left) shows the results. We can see that averaging the high quality word vectors already captures a lot of similarity. The chain structure of a standard recurrent neural net performs worst since its representation is dominated by the last words in the sequence which may not be as important as earlier words.

6.2 Image Search with Query Sentences

This experiment evaluates how well we can find images that display the visual meaning of a given sentence. We first map a query sentence into the vector space and then find images in the same space using simple inner products. As shown in Table 1 (center), the new DT-RNN outperforms all other models.

6.3 Describing Images by Finding Suitable Sentences

Lastly, we repeat the above experiments but with roles reversed. For an image, we search for suitable textual descriptions again simply by finding close-by sentence vectors in the multi-modal embedding space. Table 1 (right) shows that the DT-RNN again outperforms related models. Fig. 2 assigned to im-

<i>Image Search</i>		<i>Describing Images</i>	
Model	mRank	Model	mRank
BoW	24.7	BoW	30.7
CT-RNN	22.2	CT-RNN	29.4
Recurrent NN	28.4	Recurrent NN	31.4
kCCA	13.7	kCCA	38.0
DT-RNN	13.3	DT-RNN	26.8
SDT-RNN	15.8	SDT-RNN	37.5

Table 2: Results of multimodal ranking when models are trained with a squared error loss and using Euclidean distance in the multimodal space. Better performance is reached for all models when trained in a max-margin loss and using inner products as in the previous table.

ages. The average ranking of 25.3 for a correct sentence description is out of 500 possible sentences. A random assignment would give an average ranking of 100.

6.4 Analysis: Squared Error Loss vs. Margin Loss

We analyze the influence of the multimodal loss function on the performance. In addition, we compare using Euclidean distances instead of inner products. Table 2 shows that performance is worse for all models in this setting.

6.5 Analysis: Recall at n vs Mean Rank

Hodosh et al. (2013) and other related work use recall at n as an evaluation measure. Recall at n captures how often one of the top n closest vectors were a correct image or sentence and gives a good intuition of how a model would perform in a ranking task that presents n such results to a user. Below, we compare three commonly used and high performing models: bag of words, kCCA and our SDT-RNN on

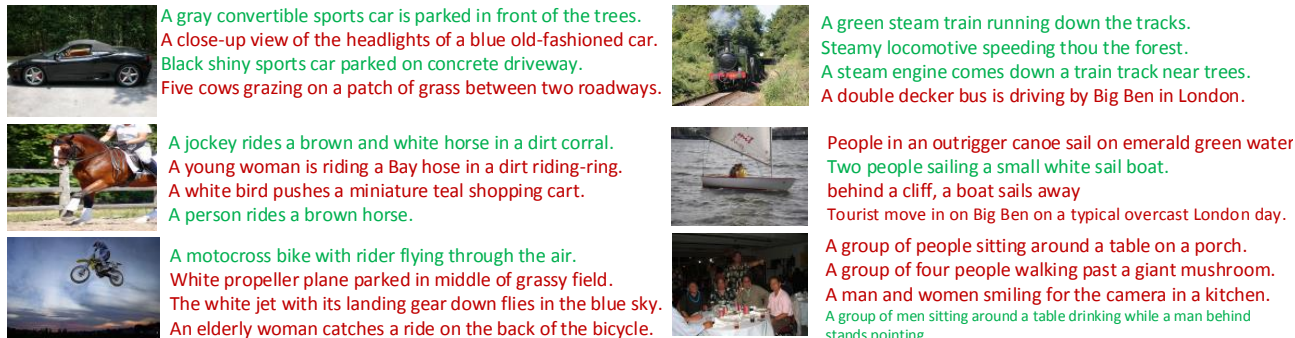


Figure 6: Images and their sentence descriptions assigned by the DT-RNN.

Model	<i>Image Search</i>			
	mRank \triangle	R@1 ∇	R@5 ∇	R@10 ∇
BoW	14.6	15.8	42.2	60.0
kCCA	15.9	16.4	41.4	58.0
SDT-RNN	12.5	16.4	46.6	65.6
Model	<i>Describing Images</i>			
	mRank \triangle	R@1 ∇	R@5 ∇	R@10 ∇
BoW	21.1	19.0	38.0	57.0
kCCA	18.0	21.0	47.0	61.0
SDT-RNN	16.9	23.0	45.0	63.0

Table 3: Evaluation comparison between mean rank of the closest correct image or sentence (lower is better \triangle) with recall at different thresholds (higher is better, ∇). With one exception (R@5, bottom table), the SDT-RNN outperforms the other two models and all other models we did not include here.

this different metric. Table 3 shows that the measures do correlate well and the SDT-RNN also performs best on the multimodal ranking tasks when evaluated with this measure.

6.6 Error Analysis

In order to understand the main problems with the composed sentence vectors, we analyze the sentences that have the worst nearest neighbor rank between each other. We find that the main failure mode of the SDT-RNN occurs when a sentence that should describe the same image does not use a verb but the other sentences of that image do include a verb. For example, the following sentence pair has vectors that are very far apart from each other even though they are supposed to describe the same image:

1. A blue and yellow airplane flying straight down while emitting white smoke
2. Airplane in dive position

Generally, as long as both sentences either have a verb or do not, the SDT-RNN is more robust to different sentence lengths than bag of words representations.

6.7 Model Analysis: Semantic Composition Matrices

The best model uses composition matrices based on semantic relationships from the dependency parser. We give some insights into what the model learns by listing the composition matrices with the largest Frobenius norms. Intuitively, these matrices have learned larger weights that are being multiplied with the child vector in the tree and hence that child will have more weight in the final composed parent vector. In decreasing order of Frobenius norm, the relationship matrices are: nominal subject, possession modifier (e.g. their), passive auxiliary, preposition at, preposition in front of, passive auxiliary, passive nominal subject, object of preposition, preposition in and preposition on.

The model learns that nouns are very important as well as their spatial prepositions and adjectives.

7 Conclusion

We introduced a new recursive neural network model that is based on dependency trees. For evaluation, we use the challenging task of mapping sentences and images into a common space for finding one from the other. Our new model outperforms baselines and other commonly used models that can compute continuous vector representations for sentences. In comparison to related models, the DT-RNN is more invariant and robust to surface changes such as word order.

References

- G. Andrew, R. Arora, K. Livescu, and J. Bilmes. 2013. Deep canonical correlation analysis. In *ICML*, Atlanta, Georgia.
- K. Barnard, P. Duygulu, N. de Freitas, D. Forsyth, D. Blei, and M. Jordan. 2003. Matching words and pictures. *JMLR*.
- M. Baroni and A. Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.
- R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167.
- F. Costa, P. Frasconi, V. Lombardo, and G. Soda. 2003. Towards incremental parsing of natural language using recursive neural networks. *Applied Intelligence*.
- M. de Marneffe, B. MacCartney, and C. D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC*.
- J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng. 2012. Large scale distributed deep networks. In *NIPS*.
- J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*.
- J. Duchi, E. Hazan, and Y. Singer. 2011. Adaptive sub-gradient methods for online learning and stochastic optimization. *JMLR*, 12, July.
- P. Duygulu, K. Barnard, N. de Freitas, and D. Forsyth. 2002. Object recognition as machine translation. In *ECCV*.
- A. Farhadi, M. Hejrati, M. A. Sadeghi, P. Young, C. Rashtchian, J. Hockenmaier, and D. Forsyth. 2010. Every picture tells a story: Generating sentences from images. In *ECCV*.
- Y. Feng and M. Lapata. 2013. Automatic caption generation for news images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35.
- A. Frome, G. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, and T. Mikolov. 2013. Devise: A deep visual-semantic embedding model. In *NIPS*.
- C. Goller and A. Küchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of the International Conference on Neural Networks*.
- E. Grefenstette, G. Dinu, Y.-Z. Zhang, M. Sadrzadeh, and M. Baroni. 2013. Multi-step regression learning for compositional distributional semantics. In *IWCS*.
- A. Gupta and L. S. Davis. 2008. Beyond nouns: Exploiting prepositions and comparative adjectives for learning visual classifiers. In *ECCV*.
- M. Hodosh, P. Young, and J. Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *J. Artif. Intell. Res. (JAIR)*, 47:853–899.
- E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng. 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. In *ACL*.
- K. Jarrett, K. Kavukcuoglu, M.A. Ranzato, and Y. LeCun. 2009. What is the best multi-stage architecture for object recognition? In *ICCV*.
- P. Blunsom, K.M. Hermann. 2013. The role of syntax in vector space models of compositional semantics. In *ACL*.
- A. Krizhevsky, I. Sutskever, and G. E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.
- G. Kulkarni, V. Premraj, S. Dhar, S. Li, Y. Choi, A. C. Berg, and T. L. Berg. 2011. Baby talk: Understanding and generating image descriptions. In *CVPR*.
- N. Kumar, A. C. Berg, P. N. Belhumeur, , and S. K. Nayar. 2009. Attribute and simile classifiers for face verification. In *ICCV*.
- P. Kuznetsova, V. Ordonez, A. C. Berg, T. L. Berg, and Yejin Choi. 2012. Collective generation of natural image descriptions. In *ACL*.
- Q. V. Le, M.A. Ranzato, R. Monga, M. Devin, K. Chen, G.S. Corrado, J. Dean, and A. Y. Ng. 2012. Building high-level features using large scale unsupervised learning. In *ICML*.
- T. Mikolov, W. Yih, and G. Zweig. 2013. Linguistic regularities in continuous spaceword representations. In *HLT-NAACL*.
- J. Mitchell and M. Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429.
- J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A.Y. Ng. 2011. Multimodal deep learning. In *ICML*.
- V. Ordonez, G. Kulkarni, and T. L. Berg. 2011. Im2text: Describing images using 1 million captioned photographs. In *NIPS*.
- J. B. Pollack. 1990. Recursive distributed representations. *Artificial Intelligence*, 46, November.
- C. Rashtchian, P. Young, M. Hodosh, and J. Hockenmaier. 2010. Collecting image annotations using Amazon’s Mechanical Turk. In *Workshop on Creating Speech and Language Data with Amazon’s MTurk*.
- R. Socher and L. Fei-Fei. 2010. Connecting modalities: Semi-supervised segmentation and annotation of images using unaligned text corpora. In *CVPR*.
- R. Socher, C. D. Manning, and A. Y. Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*.

- R. Socher, E. H. Huang, J. Pennington, A. Y. Ng, and C. D. Manning. 2011a. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *NIPS*.
- R. Socher, C. Lin, A. Y. Ng, and C.D. Manning. 2011b. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *ICML*.
- R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. 2011c. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *EMNLP*.
- R. Socher, B. Huval, C. D. Manning, and A. Y. Ng. 2012. Semantic Compositionality Through Recursive Matrix-Vector Spaces. In *EMNLP*.
- R. Socher, J. Bauer, C. D. Manning, and A. Y. Ng. 2013a. Parsing With Compositional Vector Grammars. In *ACL*.
- R. Socher, M. Ganjoo, C. D. Manning, and A. Y. Ng. 2013b. Zero-Shot Learning Through Cross-Modal Transfer. In *NIPS*.
- R. Socher, M. Ganjoo, H. Sridhar, O. Bastani, and A. Y. Ng. C. D. Manning and. 2013c. Zero-shot learning through cross-modal transfer. In *Proceedings of the International Conference on Learning Representations (ICLR, Workshop Track)*.
- N. Srivastava and R. Salakhutdinov. 2012. Multimodal learning with deep boltzmann machines. In *NIPS*.
- A. Torralba, K. P. Murphy, and W. T. Freeman. 2010. Using the forest to see the trees: exploiting context for visual object detection and localization. *Communications of the ACM*.
- P. D. Turney and P. Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.
- B. Yao, X. Yang, L. Lin, M. W. Lee, and S.-C. Zhu. 2010. I2t:image parsing to text description. *IEEE Xplore*.

Parallel Algorithms for Unsupervised Tagging

Sujith Ravi
Google
Mountain View, CA 94043
sravi@google.com

Sergei Vassilivitskii
Google
Mountain View, CA 94043
sergeiv@google.com

Vibhor Rastogi*
Twitter
San Francisco, CA
vibhor.rastogi@gmail.com

Abstract

We propose a new method for unsupervised tagging that finds minimal models which are then further improved by Expectation Maximization training. In contrast to previous approaches that rely on manually specified and multi-step heuristics for model minimization, our approach is a simple greedy approximation algorithm DMLC (DISTRIBUTED-MINIMUM-LABEL-COVER) that solves this objective in a single step.

We extend the method and show how to efficiently parallelize the algorithm on modern parallel computing platforms while preserving approximation guarantees. The new method easily scales to large data and grammar sizes, overcoming the memory bottleneck in previous approaches. We demonstrate the power of the new algorithm by evaluating on various sequence labeling tasks: Part-of-Speech tagging for multiple languages (including low-resource languages), with complete and incomplete dictionaries, and supertagging, a complex sequence labeling task, where the grammar size alone can grow to millions of entries. Our results show that for all of these settings, our method achieves state-of-the-art scalable performance that yields high quality tagging outputs.

1 Introduction

Supervised sequence labeling with large labeled training datasets is considered a solved problem. For

*The research described herein was conducted while the author was working at Google.

instance, state of the art systems obtain tagging accuracies over 97% for part-of-speech (POS) tagging on the English Penn Treebank. However, learning accurate taggers without labeled data remains a challenge. The accuracies quickly drop when faced with data from a different domain, language, or when there is very little labeled information available for training (Banko and Moore, 2004).

Recently, there has been an increasing amount of research tackling this problem using unsupervised methods. A popular approach is to learn from POS-tag dictionaries (Merialdo, 1994), where we are given a raw word sequence and a dictionary of legal tags for each word type. Learning from POS-tag dictionaries is still challenging. Complete word-tag dictionaries may not always be available for use and in every setting. When they are available, the dictionaries are often noisy, resulting in high tagging ambiguity. Furthermore, when applying taggers in new domains or different datasets, we may encounter new words that are missing from the dictionary. There have been some efforts to learn POS taggers from incomplete dictionaries by extending the dictionary to include these words using some heuristics (Toutanova and Johnson, 2008) or using other methods such as type-supervision (Garrette and Baldrige, 2012).

In this work, we tackle the problem of unsupervised sequence labeling using tag dictionaries. The first reported work on this problem was on POS tagging from Merialdo (1994). The approach involved training a standard Hidden Markov Model (HMM) using the Expectation Maximization (EM) algorithm (Dempster et al., 1977), though EM does not

perform well on this task (Johnson, 2007). More recent methods have yielded better performance than EM (see (Ravi and Knight, 2009) for an overview).

One interesting line of research introduced by Ravi and Knight (2009) explores the idea of performing model minimization followed by EM training to learn taggers. Their idea is closely related to the classic Minimum Description Length principle for model selection (Barron et al., 1998). They (1) formulate an objective function to find the smallest model that explains the text (model minimization step), and then, (2) fit the minimized model to the data (EM step). For POS tagging, this method (Ravi and Knight, 2009) yields the best performance to date; 91.6% tagging accuracy on a standard test dataset from the English Penn Treebank. The original work from (Ravi and Knight, 2009) uses an integer linear programming (ILP) formulation to find minimal models, an approach which does not scale to large datasets. Ravi et al. (2010b) introduced a two-step greedy approximation to the original objective function (called the MIN-GREEDY algorithm) that runs much faster while maintaining the high tagging performance. Garrette and Baldrige (2012) showed how to use several heuristics to further improve this algorithm (for instance, better choice of tag bigrams when breaking ties) and stack other techniques on top, such as careful initialization of HMM emission models which results in further performance gains. Their method also works under incomplete dictionary scenarios and can be applied to certain low-resource scenarios (Garrette and Baldrige, 2013) by combining model minimization with supervised training.

In this work, we propose a new scalable algorithm for performing model minimization for this task. By making an assumption on the structure of the solution, we prove that a variant of the greedy set cover algorithm always finds an approximately optimal label set. This is in contrast to previous methods that employ heuristic approaches with no guarantee on the quality of the solution. In addition, we do not have to rely on ad hoc tie-breaking procedures or careful initializations for unknown words. Finally, not only is the proposed method approximately optimal, it is also easy to distribute, allowing it to easily scale to very large datasets. We show empirically that our method, combined with an EM training step

outperforms existing state of the art systems.

1.1 Our Contributions

- We present a new method, DISTRIBUTED MINIMUM LABEL COVER, DMLC, for model minimization that uses a fast, greedy algorithm with formal approximation guarantees to the quality of the solution.
- We show how to efficiently parallelize the algorithm while preserving approximation guarantees. In contrast, existing minimization approaches cannot match the new distributed algorithm when scaling from thousands to millions or even billions of tokens.
- We show that our method easily scales to both large data and grammar sizes, and does not require the corpus or label set to fit into memory. This allows us to tackle complex tagging tasks, where the tagset consists of several thousand labels, which results in more than one million entries in the grammar.
- We demonstrate the power of the new method by evaluating under several different scenarios—POS tagging for multiple languages (including low-resource languages), with complete and incomplete dictionaries, as well as a complex sequence labeling task of supertagging. Our results show that for all these settings, our method achieves state-of-the-art performance yielding high quality taggings.

2 Related Work

Recently, there has been an increasing amount of research tackling this problem from multiple directions. Some efforts have focused on inducing POS tag clusters without any tags (Christodoulopoulos et al., 2010; Reichart et al., 2010; Moon et al., 2010), but evaluating such systems proves difficult since it is not straightforward to map the cluster labels onto gold standard tags. A more popular approach is to learn from POS-tag dictionaries (Merialdo, 1994; Ravi and Knight, 2009), incomplete dictionaries (Hasan and Ng, 2009; Garrette and Baldrige, 2012) and human-constructed dictionaries (Goldberg et al., 2008).

Another direction that has been explored in the past includes bootstrapping taggers for a new language based on information acquired from other languages (Das and Petrov, 2011) or limited annotation resources (Garrette and Baldrige, 2013). Additional work focused on building supervised taggers for noisy domains such as Twitter (Gimpel et al., 2011). While most of the relevant work in this area centers on POS tagging, there has been some work done for building taggers for more complex sequence labeling tasks such as supertagging (Ravi et al., 2010a).

Other related work include alternative methods for learning sparse models via priors in Bayesian inference (Goldwater and Griffiths, 2007) and posterior regularization (Ganchev et al., 2010). But these methods only encourage sparsity and do not explicitly seek to minimize the model size, which is the objective function used in this work. Moreover, taggers learned using model minimization have been shown to produce state-of-the-art results for the problems discussed here.

3 Model

Following Ravi and Knight (2009), we formulate the problem as that of label selection on the sentence graph. Formally, we are given a set of sequences, $\mathcal{S} = \{S_1, S_2, \dots, S_n\}$ where each S_i is a sequence of words, $S_i = w_{i1}, w_{i2}, \dots, w_{i,|S_i|}$. With each word w_{ij} we associate a set of possible tags T_{ij} . We will denote by m the total number of (possibly duplicate) words (tokens) in the corpus.

Additionally, we define two special words w_0 and w_∞ with special tags *start* and *end*, and consider the modified sequences $S'_i = w_0, S_i, w_\infty$. To simplify notation, we will refer to $w_\infty = w_{|S_i|+1}$. The sequence label problem asks us to select a valid tag $t_{ij} \in T_{ij}$ for each word w_{ij} in the input to minimize a specific objective function.

We will refer to a tag pair $(t_{i,j-1}, t_{ij})$ as a *label*. Our aim is to minimize the number of distinct labels used to cover the full input. Formally, given a sequence S'_i and a tag t_{ij} for each word w_{ij} in S'_i , let the induced set of labels for sequence S'_i be

$$L_i = \bigcup_{j=1}^{|S'_i|} \{(t_{i,j-1}, t_{ij})\}.$$

The total number of distinct labels used over all sequences is then

$$\phi = |\cup_i L_i| = \left| \bigcup_i \bigcup_{j=1}^{|S_i|+1} \{(t_{i,j-1}, t_{ij})\} \right|.$$

Note that the order of the tokens in the label makes a difference as $\{(NN, VP)\}$ and $\{(VP, NN)\}$ are two distinct labels.

Now we can define the problem formally, following (Ravi and Knight, 2009).

Problem 1 (Minimum Label Cover). *Given a set \mathcal{S} of sequences of words, where each word w_{ij} has a set of valid tags T_{ij} , the problem is to find a valid tag assignment $t_{ij} \in T_{ij}$ for each word that minimizes the number of distinct labels or tag pairs over all sequences, $\phi = \left| \bigcup_i \bigcup_{j=1}^{|S_i|+1} \{(t_{i,j-1}, t_{ij})\} \right|$.*

The problem is closely related to the classical Set Cover problem and is also NP-complete. To reduce Set Cover to the label selection problem, map each element i of the Set Cover instance to a single word sentence $S_i = w_{i1}$, and let the valid tags T_{i1} contain the names of the sets that contain element i . Consider a solution to the label selection problem; every sentence S_i is covered by two labels (w_0, k_i) and (k_i, w_∞) , for some $k_i \in T_{i1}$, which corresponds to an element i being covered by set k_i in the Set Cover instance. Thus any valid solution to the label selection problem leads to a feasible solution to the Set Cover problem $(\{k_1, k_2, \dots\})$ of exactly half the size.

Finally, we will use $\{\{\dots\}\}$ notation to denote a multiset of elements, i.e. a set where an element may appear multiple times.

4 Algorithm

In this Section, we describe the DISTRIBUTED-MINIMUM-LABEL-COVER, DMLC, algorithm for approximately solving the minimum label cover problem. We describe the algorithm in a centralized setting, and defer the distributed implementation to Section 5. Before describing the algorithm, we briefly explain the relationship of the minimum label cover problem to set cover.

4.1 Modification of Set Cover

As we pointed out earlier, the minimum label cover problem is at least as hard as the Set Cover prob-

- 1: **Input:** A set of sequences \mathcal{S} with each words w_{ij} having possible tags T_{ij} .
- 2: **Output:** A tag assignment $t_{ij} \in T_{ij}$ for each word w_{ij} approximately minimizing labels.
- 3: Let \mathcal{M} be the multi set of all possible labels generated by choosing each possible tag $t \in T_{ij}$.

$$\mathcal{M} = \bigcup_i \left(\bigcup_{j=1}^{|\mathcal{S}_i|+1} \bigcup_{\substack{t' \in T_{i,j-1} \\ t \in T_{ij}}} \{(t', t)\} \right) \quad (1)$$

- 4: Let $\mathcal{L} = \emptyset$ be the set of selected labels.
- 5: **repeat**
- 6: Select the most frequent label not yet selected: $(t', t) = \arg \max_{(s', s) \notin \mathcal{L}} |\mathcal{M} \cap (s', s)|$.
- 7: For each bigram $(w_{i,j-1}, w_{ij})$ where $t' \in T_{i,j-1}$ and $t \in T_{ij}$ tentatively assign t' to $w_{i,j-1}$ and t to w_{ij} . Add (t', t) to \mathcal{L} .
- 8: If a word gets two assignments, select one at random with equal probability.
- 9: If a bigram $(w_{i,j-1}, w_{ij})$ is consistent with assignments in (t', t) , fix the tentative assignments, and set $T_{i,j-1} = \{t'\}$ and $T_{ij} = \{t\}$. Recompute \mathcal{M} , the multi-set of possible labels, with the updated $T_{i,j-1}$ and T_{ij} .
- 10: **until** there are no unassigned words

Algorithm 1: MLC Algorithm

- 1: **Input:** A set of sequences \mathcal{S} with each words w_{ij} having possible tags T_{ij} .
- 2: **Output:** A tag assignment $t_{ij} \in T_{ij}$ for each word w_{ij} approximately minimizing labels.
- 3: (Graph Creation) Initialize each vertex v_{ij} with the set of possible tags T_{ij} and its neighbors $v_{i,j+1}$ and $v_{i,j-1}$.
- 4: **repeat**
- 5: (Message Passing) Each vertex v_{ij} sends its possibly tags T_{ij} to its forward neighbor v_{ij+1} .
- 6: (Counter Update) Each vertex receives the the tags $T_{i,j-1}$ and adds all possible labels $\{(s, s') | s \in T_{i,j-1}, s' \in T_{ij}\}$ to a global counter (M) .
- 7: (MaxLabel Selection) Each vertex queries the global counter \mathcal{M} to find the maximum label (t, t') .
- 8: (Tentative Assignment) Each vertex v_{ij} selects a tag tentatively as follows: If one of the tags t, t' is in the feasible set T_{ij} , it tentatively selects the tag.
- 9: (Random Assignment) If both are feasible it selects one at random. The vertex communicates its assignment to its neighbors.
- 10: (Confirmed Assignment) Each vertex receives the tentative assignment from its neighbors. If together with its neighbors it can match the selected label, the assignment is finalized. If the assigned tag is T , then the vertex v_{ij} sets the valid tag set T_{ij} to $\{t\}$.
- 11: **until** no unassigned vertices exist.

Algorithm 2: DMLC Implementation

lem. An additional challenge comes from the fact that labels are tags for a pair of words, and hence are related. For example, if we label a word pair $(w_{i,j-1}, w_{ij})$ as (NN, VP), then the label for the next word pair $(w_{ij}, w_{i,j+1})$ has to be of the form (VP, *), i.e., it has to start with VP.

Previous work (Ravi et al., 2010a; Ravi et al., 2010b) recognized this challenge and employed two phase heuristic approaches. Eschewing heuristics, we will show that with one natural assumption, even with this extra set of constraints, the standard greedy algorithm for this problem results in a solution with a provable approximation ratio of $O(\log m)$. In

practice, however, the algorithm performs far better than the worst case ratio, and similar to the work of (Gomes et al., 2006), we find that the greedy approach selects a cover approximately 11% worse than the optimum solution.

4.2 MLC Algorithm

We present in Algorithm 1 our MINIMUM LABEL COVER algorithm to approximately solve the minimum label cover problem. The algorithm is simple, efficient, and easy to distribute.

The algorithm chooses labels one at a time, selecting a label that covers as many words as possible in

every iteration. For this, it generates and maintains a multi-set of all possible labels \mathcal{M} (Step 3). The multi-set contains an occurrence of each valid label, for example, if $w_{i,j-1}$ has two possible valid tags NN and VP , and w_{ij} has one possible valid tag VP , then \mathcal{M} will contain two labels, namely (NN, VP) and (VP, VP) . Since \mathcal{M} is a multi-set it will contain duplicates, e.g. the label (NN, VP) will appear for each adjacent pair of words that have NN and VP as valid tags, respectively.

In each iteration, the algorithm picks a label with the most number of occurrences in \mathcal{M} and adds it to the set of chosen labels (Step 6). Intuitively, this is a greedy step to select a label that covers the most number of word pairs.

Once the algorithm picks a label (t', t) , it tries to assign as many words to tags t or t' as possible (Step 7). A word can be assigned t' if t' is a valid tag for it, and t a valid tag for the next word in sequence. Similarly, a word can be assigned t , if t is a valid tag for it, and t' a valid tag for the previous word. Some words can get both assignments, in which case we choose one tentatively at random (Step 8). If a word's tentative random tag, say t , is consistent with the choices of its adjacent words (say t' from the previous word), then the tentative choice is fixed as a permanent one. Whenever a tag is selected, the set of valid tags T_{ij} for the word is reduced to a singleton $\{t\}$. Once the set of valid tags T_{ij} changes, the multi-set \mathcal{M} of all possible labels also changes, as seen from Eq 1. The multi-set is then recomputed (Step 9) and the iterations repeated until all of words have been tagged.

We can show that under a natural assumption this simple algorithm is approximately optimal.

Assumption 1 (*c-feasibility*). *Let $c \geq 1$ be any number, and k be the size of the optimal solution to the original problem. In each iteration, the MLC algorithm fixes the tags for some words. We say that the algorithm is *c-feasible*, if after each iteration there exists some solution to the remaining problem, consistent with the chosen tags, with size at most ck .*

The assumption encodes the fact that a single bad greedy choice is not going to destroy the overall structure of the solution, and a nearly optimal solution remains. We note that this assumption of *c-feasibility* is not only sufficient, as we will formally

show, but is also necessary. Indeed, without any assumptions, once the algorithm fixes the tag for some words, an optimal label may no longer be consistent with the chosen tags, and it is not hard to find contrived examples where the size of the optimal solution doubles after each iteration of MLC.

Since the underlying problem is NP-complete, it is computationally hard to give direct evidence verifying the assumption on natural language inputs. However, on small examples we are able to show that the greedy algorithm is within a small constant factor of the optimum, specifically it is within 11% of the optimum model size for the POS tagging problem using the standard 24k dataset (Ravi and Knight, 2009). Combined with the fact that the final method outperforms state of the art approaches, this leads us to conclude that the structural assumption is well justified.

Lemma 1. *Under the assumption of *c-feasibility*, the MLC algorithm achieves a $O(c \log m)$ approximation to the minimum label cover problem, where $m = \sum_i |S_i|$ is the total number of tokens.*

Proof. To prove the Lemma we will define an objective function $\bar{\phi}$, counting the number of unlabeled word pairs, as a function of possible labels, and show that $\bar{\phi}$ decreases by a factor of $(1 - O(1/ck))$ at every iteration.

To define $\bar{\phi}$, we first define ϕ , the number of labeled word pairs. Consider a particular set of labels, $\mathcal{L} = \{L_1, L_2, \dots, L_k\}$ where each label is a pair (t_i, t_j) . Call $\{t_{ij}\}$ a valid assignment of tokens if for each w_{ij} , we have $t_{ij} \in T_{ij}$. Then the score of \mathcal{L} under an assignment t , which we denote by ϕ_t , is the number of bigram labels that appear in \mathcal{L} . Formally, $\phi_t(\mathcal{L}) = |\cup_{i,j} \{(t_{i,j-1}, t_{ij}) \cap \mathcal{L}\}|$. Finally, we define $\phi(\mathcal{L})$ to be the best such assignment, $\phi(\mathcal{L}) = \max_t \phi_t(\mathcal{L})$, and $\bar{\phi}(\mathcal{L}) = m - \phi(\mathcal{L})$ the number of uncovered labels.

Consider the label selected by the algorithm in every step. By the *c-feasibility* assumption, there exists some solution having ck labels. Thus, some label from that solution covers at least a $1/ck$ fraction of the remaining words. The selected label (t, t') maximizes the intersection with the remaining feasible labels. The conflict resolution step ensures that in expectation the realized benefit is at least a half of the maximum, thereby reducing $\bar{\phi}$ by at least a

$(1 - 1/2ck)$ fraction. Therefore, after $O(kc \log m)$ operations all of the labels are covered. \square

4.3 Fitting the Model Using EM

Once the greedy algorithm terminates and returns a minimized grammar of tag bigrams, we follow the approach of Ravi and Knight (2009) and fit the minimized model to the data using the alternating EM strategy.

In this step, we run an alternating optimization procedure iteratively in phases. In each phase, we initialize (and prune away) parameters within the two HMM components (transition or emission model) using the output from the previous phase. We initialize this procedure by restricting the transition parameters to only those tag bigrams selected in the model minimization step. We train in conjunction with the original emission model using EM algorithm which prunes away some of the emission parameters. In the next phase, we alternate the initialization by choosing the pruned emission model along with the original transition model (with full set of tag bigrams) and retrain using EM. The alternating EM iterations are terminated when the change in the size of the observed grammar (i.e., the number of unique bigrams in the tagging output) is $\leq 5\%$.¹ We refer to our entire approach using greedy minimization followed by EM training as DMLC + EM.

5 Distributed Implementation

The DMLC algorithm is directly suited towards parallelization across many machines. We turn to Pregel (Malewicz et al., 2010), and its open source version Giraph (Apa, 2013). In these systems the computation proceeds in rounds. In every round, every machine does some local processing and then sends arbitrary messages to other machines. Semantically, we think of the communication graph as fixed, and in each round each vertex performs some local computation and then sends messages to its neighbors. This mode of parallel programming directs the programmers to “Think like a vertex.”

The specific systems like Pregel and Giraph build infrastructure that ensures that the overall system

¹For more details on the alternating EM strategy and how initialization with minimized models improve EM performance in alternating iterations, refer to (Ravi and Knight, 2009).

is fault tolerant, efficient, and fast. In addition, they provide implementation of commonly used distributed data structures, such as, for example global counters. The programmer’s job is simply to specify the code that each vertex will run at every round.

We implemented the DMLC algorithm in Pregel. The implementation is straightforward and given in Algorithm 2. The multi-set \mathcal{M} of Algorithm 1 is represented as a global counter in Algorithm 2. The message passing (Step 3) and counter update (Step 4) steps update this global counter and hence perform the role of Step 3 of Algorithm 1. Step 5 selects the label with largest count, which is equivalent to the greedy label picking step 6 of Algorithm 1. Finally steps 6, 7, and 8 update the tag assignment of each vertex performing the roles of steps 7, 8, and 9, respectively, of Algorithm 1.

5.1 Speeding up the Algorithm

The implementation described above directly copies the sequential algorithm. Here we describe additional steps we took to further improve the parallel running times.

Singleton Sets: As the parallel algorithm proceeds, the set of feasible sets associated with a node slowly decreases. At some point there is only one tag that a node can take on, however this tag is rare, and so it takes a while for it to be selected using the greedy strategy. Nevertheless, if a node and one of its neighbors have only a single tag left, then it is safe to assign the unique label².

Modifying the Graph: As is often the case, the bottleneck in parallel computations is the communication. To reduce the amount of communication we reduce the graph on the fly, removing nodes and edges once they no longer play a role in the computation. This simple modification decreases the communication time in later rounds as the total size of the problem shrinks.

6 Experiments and Results

In this Section, we describe the experimental setup for various tasks, settings and compare empirical performance of our method against several existing

²We must judiciously initialize the global counter to take care of this assignment, but this is easily accomplished.

baselines. The performance results for all systems (on all tasks) are measured in terms of tagging accuracy, i.e. % of tokens from the test corpus that were labeled correctly by the system.

6.1 Part-of-Speech Tagging Task

6.1.1 Tagging Using a Complete Dictionary

Data: We use a standard test set (consisting of 24,115 word tokens from the Penn Treebank) for the POS tagging task. The tagset consists of 45 distinct tag labels and the dictionary contains 57,388 word/tag pairs derived from the entire Penn Treebank. Per-token ambiguity for the test data is about 1.5 tags/token. In addition to the standard 24k dataset, we also train and test on larger data sets—973k tokens from the Penn Treebank, 3M tokens from PTB+Europarl (Koehn, 2005) data.

Methods: We evaluate and compare performance for POS tagging using four different methods that employ the model minimization idea combined with EM training:

- EM: Training a bigram HMM model using EM algorithm (Merialdo, 1994).
- ILP + EM: Minimizing grammar size using integer linear programming, followed by EM training (Ravi and Knight, 2009).
- MIN-GREEDY + EM: Minimizing grammar size using the two-step greedy method (Ravi et al., 2010b).
- DMLC + EM: This work.

Results: Table 1 shows the results for POS tagging on English Penn Treebank data. On the smaller test datasets, all of the model minimization strategies (methods 2, 3, 4) tend to perform equally well, yielding state-of-the-art results and large improvement over standard EM. When training (and testing) on larger corpora sizes, DMLC yields the best reported performance on this task to date. A major advantage of the new method is that it can easily scale to large corpora sizes and the distributed nature of the algorithm still permits fast, efficient optimization of the global objective function. So, unlike the earlier methods (such as MIN-GREEDY) it is fast enough to run on several millions of tokens to yield additional performance gains (shown in last column).

Speedups: We also observe a significant speedup when using the parallelized version of the DMLC algorithm. Performing model minimization on the 24k tokens dataset takes 55 seconds on a single machine, whereas parallelization permits model minimization to be feasible even on large datasets. Fig 1 shows the running time for DMLC when run on a cluster of 100 machines. We vary the input data size from 1M word tokens to about 8M word tokens, while holding the resources constant. Both the algorithm and its distributed implementation in DMLC are linear time operations as evident by the plot. In fact, for comparison, we also plot a straight line passing through the first two runtimes. The straight line essentially plots runtimes corresponding to a linear speedup. DMLC clearly achieves better runtimes showing even better than linear speedup. The reason for this is that distributed version has a constant overhead for initialization, independent of the data size. While the running time for rest of the implementation is linear in data size. Thus, as the data size becomes larger, the constant overhead becomes less significant, and the distributed implementation appears to complete slightly faster as data size increases.

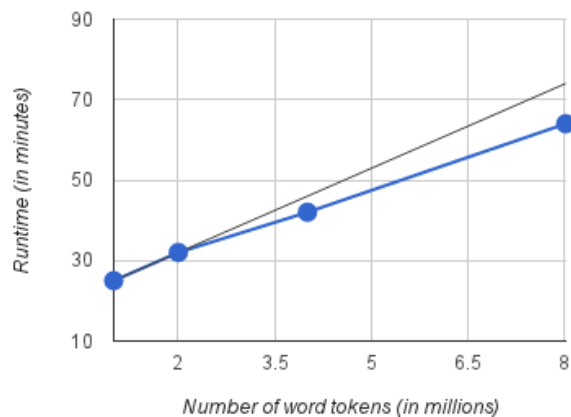


Figure 1: Runtime vs. data size (measured in # of word tokens) on 100 machines. For comparison, we also plot a straight line passing through the first two runtimes. The straight line essentially plots runtimes corresponding to a linear speedup. DMLC clearly achieves better runtimes showing a better than linear speedup.

6.1.2 Tagging Using Incomplete Dictionaries

We also evaluate our approach for POS tagging under other resource-constrained scenarios. Obtain-

Method	Tagging accuracy (%)		
	<i>te</i> =24k	<i>te</i> =973k	
	<i>tr</i> =24k	<i>tr</i> =973k	<i>tr</i> =3.7M
1. EM	81.7	82.3	
2. ILP + EM (Ravi and Knight, 2009)	91.6	-	
3. MIN-GREEDY + EM (Ravi et al., 2010b)	91.6	87.1	
4. DMLC + EM (this work)	91.4	87.5	87.8

Table 1: Results for unsupervised part-of-speech tagging on English Penn Treebank dataset. Tagging accuracies for different methods are shown on multiple datasets. *te* shows the size (number of tokens) in the test data, *tr* represents the size of the raw text used to perform model minimization.

ing a complete dictionary is often difficult, especially for new domains. To verify the utility of our method when the input dictionary is incomplete, we evaluate against standard datasets used in previous work (Garrette and Baldrige, 2012) and compare against the previous best reported performance for the same task. In all the experiments (described here and in subsequent sections), we use the following terminology—*raw* data refers to unlabeled text used by different methods (for model minimization or other unsupervised training procedures such as EM), *dictionary* consists of word/tag entries that are legal, and *test* refers to data over which tagging evaluation is performed.

English Data: For English POS tagging with incomplete dictionary, we evaluate on the Penn Treebank (Marcus et al., 1993) data. Following (Garrette and Baldrige, 2012), we extracted a word-tag dictionary from sections 00-15 (751,059 tokens) consisting of 39,087 word types, 45,331 word/tag entries, a per-type ambiguity of 1.16 yielding a per-token ambiguity of 2.21 on the raw corpus (treating unknown words as having all 45 possible tags). As in their setup, we then use the first 47,996 tokens of section 16 as raw data and perform final evaluation on the sections 22-24. We use the raw corpus along with the unlabeled test data to perform model minimization and EM training. Unknown words are allowed to have all possible tags in both these procedures.

Italian Data: The minimization strategy presented here is a general-purpose method that does not require any specific tuning and works for other languages as well. To demonstrate this, we also perform evaluation on a different language (Italian) us-

ing the TUT corpus (Bosco et al., 2000). Following (Garrette and Baldrige, 2012), we use the same data splits as their setting. We take the first half of each of the five sections to build the word-tag dictionary, the next quarter as raw data and the last quarter as test data. The dictionary was constructed from 41,000 tokens comprised of 7,814 word types, 8,370 word/tag pairs, per-type ambiguity of 1.07 and a per-token ambiguity of 1.41 on the raw data. The raw data consisted of 18,574 tokens and the test contained 18,763 tokens. We use the unlabeled corpus from the raw and test data to perform model minimization followed by unsupervised EM training.

Other Languages: In order to test the effectiveness of our method in other non-English settings, we also report the performance of our method on several other Indo-European languages using treebank data from CoNLL-X and CoNLL-2007 shared tasks on dependency parsing (Buchholz and Marsi, 2006; Nivre et al., 2007). The corpus statistics for the five languages (Danish, Greek, Italian, Portuguese and Spanish) are listed below. For each language, we construct a dictionary from the raw training data. The unlabeled corpus from the raw training and test data is used to perform model minimization followed by unsupervised EM training. As before, unknown words are allowed to have all possible tags. We report the final tagging performance on the test data and compare it to baseline EM.

Garrette and Baldrige (2012) treat unknown words (words that appear in the raw text but are missing from the dictionary) in a special manner and use several heuristics to perform better initialization for such words (for example, the probability that an unknown word is associated with a particular tag is

conditioned on the openness of the tag). They also use an auto-supervision technique to smooth counts learnt from EM onto new words encountered during testing. In contrast, we do not apply any such technique for unknown words and allow them to be mapped uniformly to all possible tags in the dictionary. For this particular set of experiments, the only difference from the Garrette and Baldrige (2012) setup is that we include unlabeled text from the test data (but without any dictionary tag labels or special heuristics) to our existing word tokens from raw text for performing model minimization. This is a standard practice used in unsupervised training scenarios (for example, Bayesian inference methods) and in general for scalable techniques where the goal is to perform inference on the same data for which one wishes to produce some structured prediction.

Language	Train (tokens)	Dict (entries)	Test (tokens)
DANISH	94386	18797	5852
GREEK	65419	12894	4804
ITALIAN	71199	14934	5096
PORTUGUESE	206678	30053	5867
SPANISH	89334	17176	5694

Results: Table 2 (column 2) compares previously reported results against our approach for English. We observe that our method obtains a huge improvement over standard EM and gets comparable results to the previous best reported scores for the same task from (Garrette and Baldrige, 2012). It is encouraging to note that the new system achieves this performance without using any of the carefully-chosen heuristics employed by the previous method. However, we do note that some of these techniques can be easily combined with our method to produce further improvements.

Table 2 (column 3) also shows results on Italian POS tagging. We observe that our method achieves significant improvements in tagging accuracy over all the baseline systems including the previous best system (+2.9%). This demonstrates that the method generalizes well to other languages and produces consistent tagging improvements over existing methods for the same task.

Results for POS tagging on CoNLL data in five different languages are displayed in Figure 2. Note that the proportion of raw data in test versus train

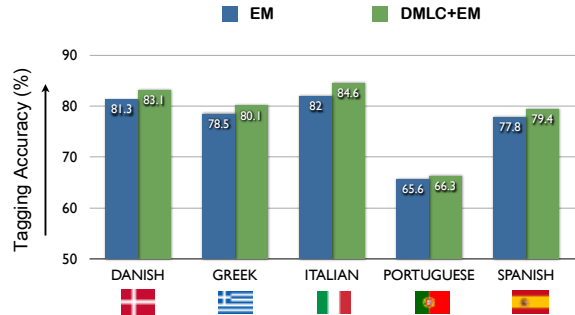


Figure 2: Part-of-Speech tagging accuracy for different languages on CoNLL data using incomplete dictionaries.

(from the standard CoNLL shared tasks) is much smaller compared to the earlier experimental settings. In general, we observe that adding more raw data for EM training improves the tagging quality (same trend observed earlier in Table 1: column 2 versus column 3). Despite this, DMLC + EM still achieves significant improvements over the baseline EM system on multiple languages (as shown in Figure 2). An additional advantage of the new method is that it can easily scale to larger corpora and it produces a much more compact grammar that can be efficiently incorporated for EM training.

6.1.3 Tagging for Low-Resource Languages

Learning part-of-speech taggers for severely low-resource languages (e.g., Malagasy) is very challenging. In addition to scarce (token-supervised) labeled resources, the tag dictionaries available for training taggers are tiny compared to other languages such as English. Garrette and Baldrige (2013) combine various supervised and semi-supervised learning algorithms into a common POS tagger training pipeline to address some of these challenges. They also report tagging accuracy improvements on low-resource languages when using the combined system over any single algorithm. Their system has four main parts, in order: (1) Tag dictionary expansion using label propagation algorithm, (2) Weighted model minimization, (3) Expectation maximization (EM) training of HMMs using auto-supervision, (4) MaxEnt Markov Model (MEMM) training. The entire procedure results in a trained tagger model that can then be applied to tag any raw data.³ Step 2 in this procedure involves

³For more details, refer (Garrette and Baldrige, 2013).

Method	Tagging accuracy (%)	
	English (PTB 00-15)	Italian (TUT)
1. Random	63.53	62.81
2. EM	69.20	60.70
3. Type-supervision + HMM initialization (Garrette and Baldrige, 2012)	88.52	72.86
4. DMLC + EM (this work)	88.11	75.79

Table 2: Part-of-Speech tagging accuracy using PTB sections 00-15 and TUT to build the tag dictionary. For comparison, we also include the results for the previously reported state-of-the-art system (method 3) for the same task.

Method	Tagging accuracy (%)		
	Total	Known	Unknown
Low-resource tagging using (Garrette and Baldrige, 2013)	80.7 (70.2)	87.6 (90.3)	66.1 (45.1)
Low-resource tagging using DMLC + EM (this work)	81.1 (70.8)	87.9 (90.3)	66.7 (46.5)

Table 3: Part-of-Speech tagging accuracy for a low-resource language (Malagasy) on *All/Known/Unknown* tokens in the test data. Tagging performance is shown for multiple experiments using different (incomplete) dictionary sizes: (a) *small*, (b) *tiny* (shown in parentheses). The new method (row 2) significantly outperforms the existing method with $p < 0.01$ for *small* dictionary and $p < 0.05$ for *tiny* dictionary.

a weighted version of model minimization which uses the multi-step greedy approach from Ravi et al. (2010b) enhanced with additional heuristics that uses tag weights learnt via label propagation (in Step 1) within the minimization process.

We replace the model minimization procedure in their Step 2 with our method (DMLC + EM) and directly compare this new system with their approach in terms of tagging accuracy. Note for all other steps in the pipeline we follow the same procedure (and run the same code) as Garrette and Baldrige (2013), including the same smoothing procedure for EM initialization in Step 3.

Data: We use the exact same setup as Garrette and Baldrige (2013) and run experiments on Malagasy, an Austronesian language spoken in Madagascar. We use the publicly available data⁴: 100k raw tokens for training, a word-tag dictionary acquired with 4 hours of human annotation effort (used for type-supervision), and a held-out test dataset (5341 tokens). We provide the unlabeled corpus from the raw training data along with the word-tag dictionary as input to model minimization and evaluate on the test corpus. We run multiple experiments for different (incomplete) dictionary scenarios: (a) *small* = 2773 word/tag pairs, (b) *tiny* = 329 word/tag pairs.

Results: Table 3 shows results on Malagasy data comparing a system that employs (unweighted)

DMLC against the existing state-of-the-art system that incorporates a multi-step weighted model minimization combined with additional heuristics. We observe that switching to the new model minimization procedure alone yields significant improvement in tagging accuracy under both dictionary scenarios. It is encouraging that a better minimization procedure also leads to higher tagging quality on the unknown word tokens (column 4 in the table), even when the input dictionary is tiny.

6.2 Supertagging

Compared to POS tagging, a more challenging task is learning supertaggers for lexicalized grammar formalisms such as Combinatory Categorical Grammar (CCG) (Steedman, 2000). For example, CCG-bank (Hockenmaier and Steedman, 2007) contains 1241 distinct supertags (lexical categories) and the most ambiguous word has 126 supertags. This provides a much more challenging starting point for the semi-supervised methods typically applied to the task. Yet, this is an important task since creating grammars and resources for CCG parsers for new domains and languages is highly labor- and knowledge-intensive.

As described earlier, our approach scales easily to large datasets as well as label sizes. To evaluate it on the supertagging task, we use the same dataset from (Ravi et al., 2010a) and compare against their baseline method that uses an modified (two-step) version

⁴[github.com/ dhgarrette/low-resource-pos-tagging-2013](https://github.com/dhgarrette/low-resource-pos-tagging-2013)

Method	Supertagging accuracy (%)	
	Ambiguous	Total
1. EM	38.7	45.6
2. ILP* + EM (Ravi et al., 2010a)	52.1	57.3
3. DMLC + EM (this work)	55.9	59.3

Table 4: Results for unsupervised supertagging with a dictionary. Here, we report the total accuracy as well as accuracy on just the ambiguous tokens (i.e., tokens which have more than one tagging possibility). *The baseline method 2 requires several pre-processing steps in order to run feasibly for this task (described in Section 6.2). In contrast, the new approach (DMLC) runs fast and also permits efficient parallelization.

of the ILP formulation for model minimization.

Data: We use the CCGbank data for this experiment. This data was created by semi-automatically converting the Penn Treebank to CCG derivations (Hockenmaier and Steedman, 2007). We use the standard splits of the data used in semi-supervised tagging experiments (Banko and Moore, 2004)—sections 0-18 for training (i.e., to construct the word-tag dictionary), and sections 22-24 for test.

Results: Table 4 compares the results for two baseline systems—standard EM (method 1), and a previously reported system using model minimization (method 2) for the same task. We observe that DMLC produces better taggings than either of these and yields significant improvement in accuracy (+2% overall, +3.8% on ambiguous tokens).

Note that it is not feasible to run the ILP-based baseline (method 2 in the table) directly since it is very slow in practice, so Ravi et al. (2010a) use a set of pre-processing steps to prune the original grammar size (unique tag pairs) from $>1M$ to several thousand entries followed by a modified two-step ILP minimization strategy. This is required to permit their model minimization step to be run in a feasible manner. On the other hand, the new approach DMLC (method 3) scales better even when the data/label sizes are large, hence it can be run with the full data using the original model minimization formulation (rather than a two-step heuristic).

Ravi et al. (2010a) also report further improvements using an alternative approach involving an ILP-based weighted minimization procedure. In Section 7 we briefly discuss how the DMLC method can be extended to this setting and combined with other similar methods.

7 Discussion and Conclusion

We present a fast, efficient model minimization algorithm for unsupervised tagging that improves upon previous two-step heuristics. We show that under a fairly natural assumption of c -feasibility the solution obtained by our minimization algorithm is $O(c \log m)$ -approximate to the optimal. Although in the case of two-step heuristics, the first step guarantees an $O(\log m)$ -approximation, the second step, which is required to get a consistent solution, can introduce many additional labels resulting in a solution arbitrarily away from the optimal. Our one step approach ensures consistency at each step of the algorithm, while the c -feasibility assumption means that the solution does not diverge too much from the optimal in each iteration.

In addition to proving approximation guarantees for the new algorithm, we show that it is parallelizable, allowing us to easily scale to larger datasets than previously explored. Our results show that the algorithm achieves state-of-the-art performance, outperforming existing methods on several different tasks (both POS tagging and supertagging) and works well even with incomplete dictionaries and extremely low-resource languages like Malagasy.

For future work, it would be interesting to apply a weighted version of the DMLC algorithm where labels (i.e., tag pairs) can have different weight distributions instead of uniform weights. Our algorithm can be extended to allow an input weight distribution to be specified for minimization. In order to initialize the weights we could use existing strategies such as grammar-informed initialization (Ravi et al., 2010a) or output distributions learnt via other methods such as label propagation (Garrette and Baldridge, 2013).

References

2013. Apache giraph. <http://giraph.apache.org/>.
- Michele Banko and Robert C. Moore. 2004. Part-of-speech tagging in context. In *Proceedings of COLING*, pages 556–561.
- Andrew R Barron, Jorma Rissanen, and Bin Yu. 1998. The Minimum Description Length Principle in Coding and Modeling. *IEEE Transactions of Information Theory*, 44(6):2743–2760.
- Cristina Bosco, Vincenzo Lombardo, Daniela Vassallo, and Leonardo Lesmo. 2000. Building a Treebank for Italian: a data-driven annotation schema. In *Proceedings of the Second International Conference on Language Resources and Evaluation LREC-2000*, pages 99–105.
- Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of CoNLL*, pages 149–164.
- Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. 2010. Two decades of unsupervised POS induction: How far have we come? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 575–584.
- Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 600–609.
- 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.
- Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11:2001–2049.
- Dan Garrette and Jason Baldridge. 2012. Type-supervised Hidden Markov Models for part-of-speech tagging with incomplete tag dictionaries. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 821–831.
- Dan Garrette and Jason Baldridge. 2013. Learning a part-of-speech tagger from two hours of annotation. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 138–147.
- Kevin Gimpel, Nathan Schneider, Brendan O’Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for Twitter: annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, pages 42–47.
- Yoav Goldberg, Meni Adler, and Michael Elhadad. 2008. EM can find pretty good HMM POS-taggers (when given a good start). In *Proceedings of ACL*, pages 746–754.
- Sharon Goldwater and Thomas L. Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *ACL*.
- Fernando C. Gomes, Cludio N. Meneses, Panos M. Pardalos, and Gerardo Valdisio R. Viana. 2006. Experimental analysis of approximation algorithms for the vertex cover and set covering problems.
- Kazi Saidul Hasan and Vincent Ng. 2009. Weakly supervised part-of-speech tagging for morphologically-rich, resource-scarce languages. In *Proceedings of the 12th Conference on the European Chapter of the Association for Computational Linguistics*, pages 363–371.
- 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.
- Mark Johnson. 2007. Why doesn’t EM find good HMM POS-taggers? In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 296–305.
- Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Machine Translation Summit X*, pages 79–86.
- Grzegorz Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. 2010. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 135–146.
- 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.
- Bernard Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–171.
- Taesun Moon, Katrin Erk, and Jason Baldridge. 2010. Crouching Dirichlet, Hidden Markov Model: Unsupervised POS tagging with context local tag generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 196–206.

- Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, pages 915–932.
- Sujith Ravi and Kevin Knight. 2009. Minimized models for unsupervised part-of-speech tagging. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP)*, pages 504–512.
- Sujith Ravi, Jason Baldrige, and Kevin Knight. 2010a. Minimized models and grammar-informed initialization for supertagging with highly ambiguous lexicons. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 495–503.
- Sujith Ravi, Ashish Vaswani, Kevin Knight, and David Chiang. 2010b. Fast, greedy model minimization for unsupervised tagging. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, pages 940–948.
- Roi Reichart, Raanan Fattal, and Ari Rappoport. 2010. Improved unsupervised POS induction using intrinsic clustering quality and a Zipfian constraint. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 57–66.
- Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA, USA.
- 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 (NIPS)*, pages 1521–1528.

Heterogeneous Networks and Their Applications: Scientometrics, Name Disambiguation, and Topic Modeling

Ben King, Rahul Jha

Department of EECS
University of Michigan
Ann Arbor, MI

{benking, rahuljha}@umich.edu

Dragomir R. Radev

Department of EECS
School of Information
University of Michigan
Ann Arbor, MI

radev@umich.edu

Abstract

We present heterogeneous networks as a way to unify lexical networks with relational data. We build a unified ACL Anthology network, tying together the citation, author collaboration, and term-cooccurrence networks with affiliation and venue relations. This representation proves to be convenient and allows problems such as name disambiguation, topic modeling, and the measurement of scientific impact to be easily solved using only this network and off-the-shelf graph algorithms.

1 Introduction

Graph-based methods have been used to great effect in NLP, on problems such as word sense disambiguation (Mihalcea, 2005), summarization (Erkan and Radev, 2004), and dependency parsing (McDonald et al., 2005). Most previous studies of networks consider networks with only a single type of node, and in some cases using a network with a single type of node can be an oversimplified view if it ignores other types of relationships.

In this paper we will demonstrate *heterogeneous networks*, networks with multiple different types of nodes and edges, along with several applications of them. The applications in this paper are not presented so much as robust attempts to out-perform the current state-of-the-art, but rather attempts at being competitive against top methods with little effort beyond the construction of the heterogeneous network.

Throughout this paper, we will use the data from the ACL Anthology Network (AAN) (Bird et al., 2008; Radev et al., 2013), which contains additional metadata relationships not found in the ACL Anthology, as a typical heterogeneous network. The results

in this paper should be generally applicable to other heterogeneous networks.

1.1 Heterogeneous AAN schema

We build a heterogeneous graph $G(V, E)$ from AAN, where V is the set of vertices and E is the set of edges connecting vertices. A vertex can be one of five semantic types: {paper, author, venue, institution, term}. An edge can also be one of five types, each connecting different types of vertices:

- author — [writes] — paper
- paper — [cites] — paper
- paper — [published in] — venue¹
- author — [affiliated with] — institution²
- paper — [contains] — term

All of this data, except for the terms, is available for all papers in the 2009 release of AAN. Terms are extracted from titles by running TextRank (Mihalcea and Tarau, 2004) on NP-chunks from titles and manually filtering out bad terms.

We show the usefulness of this representation in several applications: the measurement of scientific impact (Section 2), name disambiguation (Section 3), and topic modeling (Section 4). The heterogeneous network representation provides a simple framework for combining lexical networks (like the term co-occurrence network) with metadata relations from a source like AAN and allows us to begin to develop NLP-aware methods for problems like scientometrics and name disambiguation, which are not usually framed in an NLP perspective.

¹For a joint meeting of venues A and B publishing a paper x , two edges (x, A) and (x, B) are created.

²Author-affiliation edges are weighted according to the number of papers an author has published from an institution.

2 Scientific Impact Measurement

The study of scientometrics, which attempts to quantify the scientific impact of papers, authors, etc. has received much attention recently, even within the NLP community. In the past few years, there have been many proposed measures of scientific impact based on relationships between entities. Intuitively, a model that can take into account many different types of relationships between entities should be able to measure scientific impact more accurately than simpler measures like citation counts or h-index.

We propose using *Pagerank on the heterogeneous AAN* (Page et al., 1999) to measure scientific impact. Since changes in the network schema can affect the relative rankings between different types of entities, this method is probably not appropriate for comparing entities of two different types against each other. But between nodes of the same type, this measure is an appropriate (and as we will show, accurate) way to compare impacts.

We see this method as a first logical step in the direction of heterogeneous network-based scientometrics. This method could easily be extended to use a directed schema (Kurland and Lee, 2005) or a schema that is aware of the lexical content of citation sentences, such as sentiment-based signed networks (Hassan et al., 2012).

Determining the intrinsic quality of scientific impact measures can be difficult since there is no way to collect gold standard measurements for real-world entities. Previous studies have attempted to show that their measures give high scores to a few known high-impact entities, *e.g.* Nobel prize winners (Hirsch, 2005), or have performed a statistical component analysis to find the most important measures in a group of related statistics (Bollen et al., 2009). Our approach, instead, is to generate *realistic* data from *synthetic* entities whose impacts are known.

We had considered alternative formulations that did not rely on synthetic data, but each of them presented problems. When we attempted manual prominence annotation for AAN data, the inter-judge agreement (measured by Spearman correlation) in our experiments ranged from decent (0.9 in the case of institutions) to poor (0.3 for authors)

to nearly random (0.03 for terms), far too low to use in most cases. We also considered evaluating prominence measures by their ability to predict future citations to an entity. Citations are often used as a proxy for impact, but our measurements have found that correlation between past citations and future citations is too high for citation prediction to be a meaningful evaluation³.

2.1 Creating a synthetic AAN

In network theory, a common technique for testing network algorithms when judgments of real-world data are expensive or impossible to obtain is to test the algorithm on a synthetic network. To create such a synthetic network, the authors define a simple, but realistic generative process by which the real-world networks of interest may arise. The properties of the network are measured to ensure that it replicates certain observable behaviors of the real-world network. They can then test network algorithms to see how well they are able to recover the hidden parameters that generated the synthetic network. (Pastor-Satorras and Vespignani, 2001; Clauset et al., 2009; Karrer and Newman, 2011)

We take a two-step approach to generating this synthetic data, first generating entities with known impacts, and second, linking these entities together according to their latent impacts. Our heuristic is that high impact entities should be linked to other high impact entities and vice-versa. As in the network theory literature, we must show that this data reflects important properties observed in the true AAN.

One such property is that the number of citations per paper follows a power law distribution (Redner, 1998). We observe this behavior in AAN along with several other small-world behaviors, such as a small diameter, a small average shortest path length, and a high clustering coefficient in the coauthorship graph. We strive to replicate these properties in our synthetic data.

³Most existing impact measurements require access to at least one year's worth of citation information. The Spearman correlation between the number of citations received after one year and after five years is 0.79 with correlation between successive years as high as 0.99. Practically this means that the measures that best correlate with citations after five years are exactly those that best correlate with citations after one year.

Since scientific impact measures attempt to quantify the true impact of entities, we can use these measures to help understand how the true impact measures are distributed across different entities. In fact, citation counts, being a good estimate of impact, can be used to generate these latent impact variables for each entity. For each type of entity (papers, authors, institutions, venues, and terms), we create a latent impact by sampling from the appropriate citation count distribution. After sampling, all the impacts are normalized to fall in the $[0, 1]$ interval, with the highest-impact entity of each type having a latent impact of 1. Additive smoothing is used to avoid having an impact of 0.

Once we have created the entities, our method for placing edges is most similar to the Erdős-Rényi method for creating random graphs (Erdős and Rényi, 1960), in which edges are distributed uniformly at random between pairs of vertices. Instead of distributing links uniformly, links between entities are sampled proportionally to $I(a)I(b)(1 - (I(a) - I(b))^2)$, where $I(x)$ is the latent impact of entity x .

We tried several other formulae that failed to replicate the properties of the real AAN. The $I(a)I(b)$ part of the formula above reflects a preference for nodes of any type to connect with high-impact entities (*e.g.*, major conferences receive many submissions even though most submissions will be rejected), but the $1 - (I(a) - I(b))^2$ part also reflects the reality that entities of similar prominence are most likely to attach to each other (*e.g.*, well-known authors publish in major conferences, while less well-known authors may publish mostly in lesser-known workshops).

Using this distribution, we randomly sample links between papers and authors; authors and institutions; papers and venues; and papers and terms. The only exception to this was paper-to-paper citation links, for which we did not expect this same behavior to apply, as low-impact papers regularly cite high-impact papers, but not *vice-versa*. To model citations, we selected citing papers uniformly at random and cited papers in proportion to their impacts. (Albert and Barabási, 2002)

Finally, we generated a network equal in size to AAN, that is, with the exact same numbers of papers, authors, etc. and the exact same number of

Relationship	True value	Synth. value
Paper-citations power law coeff.	1.82	2.12
Diameter	9	8
Avg. shortest path	4.27	4.05
Collaboration network clustering coeff.	0.34	0.26

Table 1: Network properties of the synthetic AAN compared with the true AAN.

paper-author links, paper-venue links, etc. Table 1 compares the observed properties of the true AAN with the observed properties of this synthetic version of AAN. None of the statistics are exact matches, but when building random graphs, it is not uncommon for measures to differ by many orders of magnitude, so a model that has measures that are on the same order of magnitude as the observed data is generally considered to be a decent model (Newman and Park, 2003).

2.2 Measuring impact on the synthetic AAN

This random network is, of course, still imperfect in some regards. First of all, it has no time aspect, so it is not possible for impact to change over time, which means we cannot test against some impact measures that have a time component like CiteRank (Maslov and Redner, 2008). Second, there are some constraints present in the real world that are not enforced here. Because the edges are randomly selected, some papers have no venues, while others have multiple venues. There is also nothing to enforce certain consistencies, such as authors publishing many papers from relatively few institutions, or repeatedly collaborating with the same authors.

We had also considered using existing random graph models such as the Barabási-Albert model (Barabási and Albert, 1999), which are known to produce graphs that exhibit power law behavior. These models, however, do not provide a way to respect the latent impacts of the entities, as they add links in proportion only to the number of existing links a node has.

We measure the quality of impact measures by comparing ranked lists: the ordering of the entities

Paper measure	Agreement
Heterogeneous network Pagerank	0.773
Citation network Pagerank	0.558
Citation count	0.642
Author measure	Agreement
Heterogeneous network Pagerank	0.461
Coauthorship network Pagerank	0.244
h-index (Hirsch, 2005)	0.292
Aggregated citation count	0.236
i10-index	0.235
Institution measure	Agreement
Heterogeneous network Pagerank	0.373
h-index (Mitra, 2006)	0.334
Aggregated citation count	0.327
Venue measure	Agreement
Heterogeneous network Pagerank	0.449
h-index (Braun et al., 2006)	0.425
Aggregated citation count	0.370
Impact factor	0.092
Venue citation network Pagerank (Bollen et al., 2006)	0.366

Table 2: Agreement of various impact measures with the true latent impact.

by their true (but hidden) impact against their ordering according to the impact measure. The agreement between these lists is measured by Kendall’s Tau. Table 2 compares several well-known impact measures with our impact measure, Pagerank centrality on the heterogeneous AAN network. We find that some popular methods, such as h-index (Hirsch, 2005) are too coarse to accurately capture much of the underlying variation. There is a version of Kendall’s Tau that accounts for ties, and while this metric slightly helps the coarser measures, Pagerank on the heterogeneous network is still the clear winner.

When comparing different ordering methods, it is natural to wonder which of entities the orderings disagree on. In general, non-heterogeneous measures like h-index or collaboration network Pagerank, which only focus on one type of relationship can suffer when the entity in question has an important relationship of another type. For example, if an author is highly cited, but mostly works alone, his

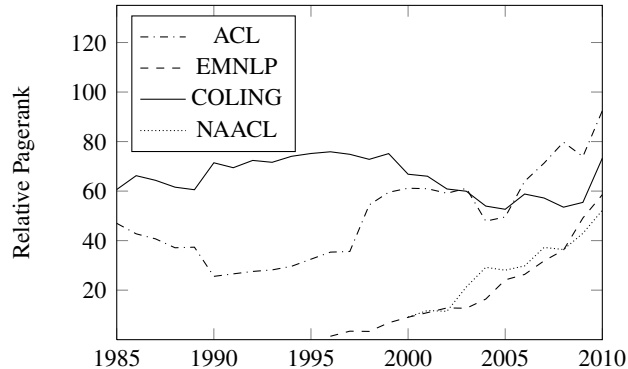


Figure 1: Evolution of conference impacts. The y -axis measures relative Pagerank, the entity’s Pagerank relative to the average Pagerank in that year.

contribution would be undervalued in the collaboration network, but would be more accurate in the heterogeneous network.

The majority of the differences between the impact measures, though, tend to be in how they handle entities of low prominence. It seems that, for the most part, there is relatively little disagreement in the orderings of high-impact entities between different impact measures. That is, most highly prominent entities tend to be highly rated by most measures. But when an author or a paper, for example, only has one or two citations, it can be advantageous to look at more types of relationships than just citations. The paper may be written by an otherwise prominent author, or published at a well-known venue, and having many types of relations at its disposal can help a method like heterogeneous network Pagerank better distinguish between two low-prominence entities.

2.3 Top-ranked entities according to heterogeneous network PageRank

Table 3 shows the papers, authors, institutions, venues, and terms that received the highest Pagerank in the heterogeneous AAN. It is obvious that the top-ranked entities in this network are not simply the most highly cited entities.

This ranking also does not have any time bias toward the entities that are currently prominent, as some of the top authors were more prolific in previous decades than at the current time. We also see this effect with COLING, which for many of the early years, is the only venue in the ACL Anthology.

Top Papers	Top Authors	Top Institutions	Top Venues	Top Terms
– Building A Large Annotated Corpus Of English: The Penn Treebank	△ 15 Jun’ichi Tsujii	△ 8 Carnegie Mellon University	△ 1 COLING	– translation
– The Mathematics Of Statistical Machine Translation: Parameter Estimation	△ 7 Aravind K. Joshi	△ 1 University of Edinburgh	▽ 1 ACL	△ 3 speech
– Attention, Intentions, And The Structure Of Discourse	△ 18 Ralph Grishman	▽ 2 University of Pennsylvania	△ 2 HLT	▽ 1 parsing
– A Maximum Entropy Approach To Natural Language Processing	△ 75 Hitoshi Isahara	▽ 2 Massachusetts Institute of Technology	△ 4 EACL	▽ 1 machine translation
– BLEU: a Method for Automatic Evaluation of Machine Translation	△ 20 Yuji Matsumoto	△ 12 Saarland University	△ 7 LREC	△ 3 generation
– A Maximum-Entropy-Inspired Parser	△ 7 Kathleen R. McKeown	▽ 2 IBM T.J. Watson Research Center	– NAACL	△ 3 evaluation
△ 2 A Stochastic Parts Program And Noun Phrase Parser For Unrestricted Text	△ 13 Eduard Hovy	△ 39 CNRS	▽ 3 EMNLP	△ 6 grammar
▽ 1 A Systematic Comparison of Various Statistical Alignment Models	△ 10 Christopher D. Manning	△ 26 University of Tokyo	▽ 5 Computational Linguistics	△ 16 dialogue
△ 4 Transformation-Based Error-Driven Learning and Natural Language Processing: a Case Study in Part-of-Speech Tagging	△ 93 Yorrick Wilks	▽ 4 Stanford University	△ 4 IJCNLP	△ 10 knowledge
△ 1 A Maximum Entropy Model for Part-of-Speech Tagging	▽ 9 Hermann Ney	△ 3 BBN Technologies	△ 1 Workshop on Speech and Natural Language	△ 1 discourse

Table 3: The entities of each type receiving the highest scores from the heterogeneous network Pagerank impact measure along with their respective changes in ranking when compared to a simple citation count measure.

One possible way to address this is to use a narrower time window when creating the graph, such as only including edges from the previous five years. We apply this technique in the following section.

2.4 Entity impact evolution

The heterogeneous graph formalism also provides a natural way to study the *evolution of impact* over time, as in (Hall et al., 2008), but at a much finer granularity. Hall et al. measured the year-by-year prominence of statistical topics, but we can measure year-by-year prominence for any entity in the graph.

To measure the evolution of impacts over the years, we iteratively create year-by-year versions of the heterogeneous AAN. Each of these graphs contains all entities along with all edges occurring in a five year window. Due to space, we cannot comprehensively exhibit this technique and the data it produces, but as a brief example, in Figure 1, we show how the impacts of some major NLP conferences changes over time.

The graph shows that NAACL and EMNLP have been steadily gaining prominence since their intro-

ductions, but also shows that ACL has had to make up a lot of ground since 1990 to surpass COLING. We also notice that all the major conferences have grown in impact since 2005, and believe that as the field continues to grow, the major conferences will continue to become more and more important.

3 Name Disambiguation

We frame network name disambiguation in a link prediction setting (Taskar et al., 2003; Liben-Nowell and Kleinberg, 2007). The problems of name disambiguation and link prediction share many characteristics, and we have found that if two ambiguous name nodes are close enough to be selected by a link-prediction method, then they likely correspond to the same real-world author.

We intend to show that the heterogeneous bibliographic network can be used to better disambiguate author names than the author collaboration network. The heterogeneous network for this problem contains papers, authors, terms, venues, and institutions. We compare several well-known network similarity measures from link prediction by transforming the

Network	Distance Measure	Precision	Recall	F1-score	Rand index	Purity	NMI
Heterogeneous	Truncated Commute Time	0.59	0.78	0.63	0.63	0.71	0.43
Heterogeneous	Shortest Path	0.90	0.79	0.83	0.87	0.94	0.76
Heterogeneous	PropFlow	0.89	0.83	0.84	0.87	0.93	0.77
Coauthorship	Truncated Commute Time	0.47	0.80	0.54	0.47	0.60	0.18
Coauthorship	Shortest Path	0.54	0.73	0.60	0.61	0.67	0.31
Coauthorship	PropFlow	0.57	0.76	0.64	0.66	0.71	0.43
Coauthorship	GHOST	0.89	0.60	0.69	0.81	0.94	0.63

Table 4: Performance of different networks and distance measures on the author name disambiguation task. The performance measures are averaged over the sets of two, three, and four authors. Rand index is from (Rand, 1971) and NMI is an abbreviation for normalized mutual information (Strehl and Ghosh, 2003)

similarities to distances and inducing clusters of authors based on these distances.

We compare three distance measures: shortest path, truncated commute time (Sarkar et al., 2008), and PropFlow (Lichtenwalter et al., 2010). *Shortest path distance* can be a useful metric for author disambiguation because it is small when two ambiguous nodes are neighbors in the graph or share a neighbor. Its downside is that it only considers one path between nodes, the shortest, and cannot take advantage of the fact that there may be many short paths between two nodes.

Truncated commute time is a variant of commute time where all paths longer than some threshold are truncated. The truncation threshold l should be set such that no semantically meaningful path is truncated. We use a value of ten for l in the heterogeneous graph and three in the coauthorship graph⁴. The advantage of truncated commute time over ordinary commute time is simpler calculation, as no paths longer than l need be considered. The downside of this method is that large branching factors tend to lead to less agreement between commute time and truncated commute time.

PropFlow is a quantity that measures the probability that a non-intersecting random walk starting at node a reaches node b in l steps or fewer, where l is again a threshold. As before, l should be a bound on the length of semantically meaningful paths, so we use the same values for l as with truncated commute time. Of course, PropFlow is not a metric, which is

⁴This is a standard coauthorship graph with the edge weights equal to the number of publications shared between authors. The heterogeneous network does not have author-to-author links, as authors are linked by paper nodes.

required for some clustering methods. We use the following equation to transform PropFlow to a metric: $d(a, b) = \frac{1}{PropFlow(a,b)} - 1$.

With each of the distance measures, we apply the same clustering method: partitioning around medoids, with the number of clusters automatically determined using the gap statistic method (Tibshirani et al., 2001). We create the null distribution needed for the gap statistic method by many iterations of randomly sampling distances from the complete distance matrix between all nodes in the graph. The gap statistic method automatically selects the number of clusters from two, three, or four author clusters.

We compare our methods against GHOST (Fan et al., 2011), a high-performance author disambiguation method based on the coauthorship graph.

3.1 Data

To generate name disambiguation data, we use the *pseudoword method* of (Gale et al., 1992). Specifically, we choose two or more completely random authors and conflate them by giving all instances of both authors the same name. We let each paper written by this pseudoauthor be an instance to be clustered. The clusters produced by any author disambiguation method can then be compared against the papers actually written by each of the two authors. This method, of course, relies on having all of the underlying authors completely disambiguated, which AAN provides.

This method is used to create 100 distambiguation sets with two authors, 100 for three authors, and 100 for four authors.

3.2 Results

Table 4 shows the performance of author name disambiguation with different networks and distance metrics. F1-score is the measure that is most often used to compare author disambiguation methods. Both PropFlow and shortest path similarity on the heterogeneous network perform quite well according to this measure, as well as the other reported measures. While comparable recall can be achieved using only the coauthorship graph, the heterogeneous graph allows for much higher precision.

4 Random walk topic model

Here we present a topic model based entirely on graph random walks. This method is not truly a statistical model as there are no statistical parameters being learned, but rather a topic-discovery and -assignment method, attempting to solve the same problem as statistical topic models such as probabilistic latent semantic analysis (pLSA) (Hofmann, 1999) or latent Dirichlet allocation (LDA) (Blei et al., 2003). In the absence of better terminology, we use the name *random walk topic model*.

While this method does not have the robust mathematical foundation that statistical topic models possess, in its favor it has modularity, simplicity, and interpretability. This language model is *modular* as it completely separates the discovery of topics from the association of topics with entities. It is *simple* because it requires only a clustering algorithm and random walk algorithms, instead of complex inference algorithms. The method also does not require any modification if the topology of the network changes, whereas statistical models may need an entirely different inference procedure if, *e.g.*, author topics are desired in addition to paper topics. Thirdly this method is easily *interpretable* with topics provided by clustering in the word-relatedness graph and topic association based on random walks from entities to topics.

4.1 Topics from word graph clustering

From the set of ACL anthology titles, we create two graphs: (1) a *word relatedness graph* by creating a weighted link between each pair of words corresponding to the PropFlow (Lichtenwalter et al., 2010) measure between them on the full heteroge-

neous graph and (2) a *word co-occurrence graph* by creating a weighted link between each pair of words corresponding to the number of titles in which both words occur.

Both of these graphs are then clustered using Graph Factorization Clustering (GFC). GFC is a soft clustering algorithm for graphs that models graph edges as a mixture of latent node-cluster association variables. (Yu et al., 2006)

Given a word graph G with vertices V and adjacency matrix $[w]_{ij}$, GFC attempts to fit a bipartite graph $K(V, U)$ with adjacency matrix $[b]_{ij}$ onto this data, with the m nodes of U representing the clusters. Whereas in G , similarity between two words i and j can be measured with w_{ij} , we can similarly measure their similarity in K with $w'_{ij} = \sum_{p=1}^m \frac{b_{ip}b_{jp}}{\lambda_p}$ where $\lambda_p = \sum_{i=1}^n b_{ip}$ is the degree of vertex $p \in U$.

Essentially the bipartite graph attempts to approximate the transition probability between i and j in G with the sum of transition probabilities from i to j through any of the m nodes in U . Yu, et al. (2006) present an algorithm for minimizing the divergence distance $\ell(\mathbf{X}, \mathbf{Y}) = \sum_{ij} (x_{ij} \log \frac{x_{ij}}{y_{ij}} - x_{ij} + y_{ij})$ between $[w]_{ij}$ and $[w']_{ij}$.

We run GFC with this distance metric and $m = 100$ clusters on the word graph until convergence (change in log-likelihood $< 0.1\%$). After convergence, the nodes in U become the clusters and the weights b_{ip} (constrained to sum to 1 for each cluster) become the topic-word association scores.

Examples of some topics found by this method are shown in Table 5. From manual inspection of these topics, we found them to be very much like topics created by statistical topic models. We find instances of all the types of topics listed in (Mimno et al., 2011): chained, intruded, random, and unbalanced. For an evaluation of these topics see Section 4.3.1.

4.2 Entity-topic association

To associate entities with topics, we first create the heterogeneous network as in previous sections, adding links between papers and their title words, along with links between words and the topics that were discovered in the previous section. Word-topic links are also weighted according to the weights

Word sense induction	sense disambiguation word induction unsupervised clustering senses based similarity chinese
CRFs + their applications	entity named recognition random conditional fields chinese entities biomedical segmentation
Dependency parsing	parsing dependency projective probabilistic incremental deterministic algorithm data syntactic trees
Tagging	models tagging model latent markov conditional random parsing unsupervised segmentation
Multi-doc summarization	summarization multi document text topic based query extractive focused summaries
Chinese word segmentation	word segmentation chinese based alignment character tagging bakeoff model crf
Lexical semantics	lexical semantic distributional similarity wordnet resources lexicon acquisition semantics representation
Cross-lingual IR	cross lingual retrieval document language linguistic multi person multilingual coreference
Generation for summar.	sentence based compression text summarization ordering approach ranking generation
Spoken language	speech recognition automatic prosodic tagging spontaneous news broadcast understanding conversational
French function words	de la du des le automatique analyse une en pour
Question answering	question answering system answer domain retrieval web based open systems
Unsupervised learning	unsupervised discovery learning induction knowledge graph acquisition concept clustering pattern
SVMs for NLP	support vector machines errors space classification correcting word parsing detecting
MaxEnt models	entropy maximum approach based attachment model models phrase prepositional disambiguation
Dialogue systems	dialogue spoken systems human conversational multi interaction dialogues utterances multimodal
Semantic role-labeling	semantic role labeling parsing syntactic features ill dependency formed framenet
SMT	based translation machine statistical phrase english approach learning reordering model
Coreference resolution	resolution coreference anaphora reference pronoun ellipsis ambiguity resolving approach pronominal
Semi- and weak-supervision	learning supervised semi classification active data clustering approach graph weakly
Information retrieval	based retrieval similarity models semantic space model distance measures document
Discourse	discourse relations structure rhetorical coherence temporal representation text connectives theory
CFG parsing	context free grammars parsing linear probabilistic rewriting grammar systems optimal
Min. risk train. and decod.	minimum efficient training error rate translation risk bayes decoding statistical
Phonology	phoneme conversion letter phonological grapheme rules applying transliteration syllable sound
Sentiment	sentiment opinion reviews classification mining polarity analysis predicting product features
Neural net speech recog.	speech robust recognition real network time neural networks language environments
Finite state methods	state finite transducers automata weighted translation parsing incremental minimal construction
Mechanical Turk	mechanical turk automatic evaluation amazon techniques data articles image scientific

Table 5: Top 10 words for several topics created by the co-occurrence random walk topic model. The left column is a manual label.

Topic 59		Topic 82	
translation	0.1953	parsing	0.1715
machine	0.1802	dependency	0.1192
statistical	0.0784	projective	0.0138
Machine Translation	0.0018	K-best Spanning Tree Parsing	0.0025
Better Hypothesis Testing for Statistical	0.0016	Pseudo-Projective Dependency Parsing	0.0024
Machine Translation: Controlling for			
Optimizer Instability			
Filtering Antonymous, Trend- Contrasting, and	0.0015	Shift-Reduce Dependency DAG Parsing	0.0017
Polarity-Dissimilar Distributional Paraphrases			
for Improving Statistical Machine Translation			
Knight, Kevin	0.0083	Nivre, Joakim	0.0120
Koehn, Philipp	0.0074	Johnson, Mark	0.0085
Ney, Hermann	0.0072	Nederhof, Mark-Jan	0.0064
RWTH Aachen University	0.0212	Vaxjo University	0.0113
Carnegie Mellon University	0.0183	Brown University	0.0107
University of Southern California	0.0177	University of Amsterdam	0.0094
Workshop on Statistical Machine Translation	0.0590	ACL	0.0512
EMNLP	0.0270	EMNLP	0.0259
COLING	0.0173	CoNLL	0.0223

Table 6: Examples of entities associated with selected topics.

determined by GCF. We then simply take random walks from topics to entities and measure the proportion at which the random walk arrives at each entity of interest. These proportions become the entity-topic association scores.

For example, if we wanted to find the authors most associated with topic 12, we would take a number of random walks (say 50,000) starting at topic 12 and terminating as soon as the random walk first reaches an author node. Measuring the proportion at which random walks arrive at each allows us to compute an association score between topic 12 and each author.

A common problem in random walks on large graphs is that the walk can easily get “lost” between two nodes that should be very near by taking a just a few steps in the wrong direction. To keep the random walks from taking these wrong steps, we adjust the topology of the network using directed links to keep the random walks moving in the “right” direction. We design the graph such that if we desire a random walk from nodes of type s to nodes of type t , the random walk will never be able to follow an outgoing link that does not decrease its distance from the nodes of t .

As shown in section 2.3, there are certain nodes at which a random walk (like Pagerank) arrives at more often than others simply because of their positions in the graph. This suggests that there may be stationary random walk distributions over entities, which we would need to adjust for in order to find the most *significant* entities for a topic.

Indeed this is what we do find. As an example, if we sample topics uniformly and take random walks to author nodes, by chance we end up at Jun’ichi Tsujii on 0.3% of random walks, Eduard Hovy on 0.2% of walks, etc. These values are about 1000 times greater than would be expected at random.

To adjust for this effect, when we take a random walk from a topic x to an entity type t , we subtract out this stationary distribution for t , which corresponds to the proportion of random walks that end at any particular entity of type t by chance, and not by virtue of the fact that the walk started at topic x . The resulting distribution yields the entities of t that are most significantly associated with topic x . Table 6 gives examples of the most significant entities for a couple of topics.

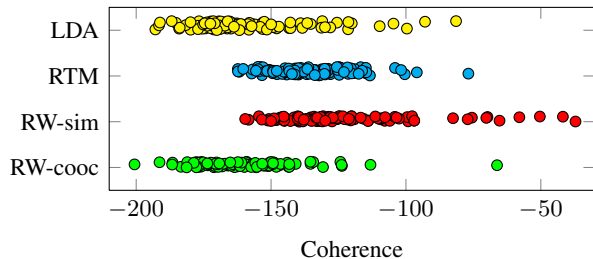


Figure 2: Distribution of topic coherences for the four topic models.

4.3 Topic Model Evaluation

We provide two separate evaluations in this section, one of the topics alone, and one extrinsic evaluation of the entire paper-topic model. The variants of random walk topic models are compared against LDA and the relational topic model (RTM), each with 100 topics (Chang and Blei, 2010). As RTM allows only a single type of relationship between documents, we use citations as the inter-document relationships.

4.3.1 Topic Coherence

The *coherence* of a topic is evaluated using the coherence metric introduced in (Mimno et al., 2011). Given the top M words $V^{(t)} = (v_1^{(t)}, \dots, v_M^{(t)})$ for a topic t , the coherence of that topic can be calculated with the following formula:

$$C(t; V^{(t)}) = \sum_{m=2}^M \sum_{l=1}^{m-1} \log \left(\frac{D(v_m^{(t)}, v_l^{(t)}) + 1}{D(v_l^{(t)})} \right),$$

where $D(v)$ is the number of documents containing v and $D(v, v')$ is the number of documents containing both v and v' .

This measure of coherence is highly correlated with manual annotations of topic quality, with a higher coherence score corresponding to a more coherent, higher quality topic. After calculating the coherence for each of the 100 topics for RTM and the random-walk topic model, the average coherence for RTM topics was -135.2 and the average coherence for word-similarity random walk topics was -122.2, with statistical significance at $p < 0.01$. Figure 2 demonstrates this, showing that the word similarity-based random walk method generates several highly coherent topics. The average coherence for the LDA and the co-occurrence random walk model were significantly lower.

4.3.2 Extrinsic Evaluation

One difficulty in evaluating this random-walk topic model intrinsically against a statistical topic model like RTM is that existing evaluation measures assume certain statistical properties of the topic, for example, that the topics are generated according to a Dirichlet prior. Because of this, we choose instead to evaluate this topic model extrinsically with a downstream application. We choose an information retrieval application, returning a ranked list of similar documents, given a reference document.

We evaluate five different methods: citation-RTM, LDA, the two versions of the random-walk topic model, and a simple word vector similarity baseline. Similarity between documents with the topic models are determined by cosine similarity between the topic vectors of the two documents. Word vector similarity determines the similarity between documents by taking the cosine similarity of their word vectors. From these similarity scores, a ranked list is produced.

The document set for this task is the set of all papers appearing at ACL between 2000 and 2011. The top 10 results returned by each method are pooled and manually evaluated with a relevance score between 1 and 10. Thirty such result sets were manually annotated. We then evaluate each method according to its discounted cumulative gain (DCG) (Järvelin and Kekäläinen, 2000).

Performance of these methods is summarized in Table 7. The co-occurrence-based random walk topic model performed comparably with the best performer at this task, LDA, and there was no significant difference between the two at $p < 0.05$.

Going forward, an important problem is to reconcile the co-occurrence- and word-similarity-based formulations of this topic model, as the two formulations perform very differently in our two evaluations. Heuristically, the co-occurrence model seems to create good human-readable topics, while the word-similarity model creates topics that are more mathematically-coherent, but less human-readable.

5 Related Work

Heterogeneous networks have been studied in a number of different fields, such as biology (Sioson, 2005), transportation networks (Lozano and

Method	DCG
Word vector	1.345 ± 0.007
LDA	3.302 ± 0.008
RTM	3.058 ± 0.011
Random-walk (cooc)	3.295 ± 0.006
Random-walk (sim)	2.761 ± 0.007

Table 7: DCG Performance of the various topic models and baselines on the related document finding task. A 95% confidence interval is provided.

Storchi, 2002), social networks (Lambiotte and Ausloos, 2006), and bibliographic networks (Sun et al., 2011). These networks are also sometimes known by the name complex networks or multimodal networks, but both these terms have other connotations. We prefer “heterogeneous networks” as used by Sun et al. (2009).

There has also been some study of these networks in general, in community detection (Murata, 2010), clustering (Long et al., 2008; Sun et al., 2012), and data mining (Muthukrishnan et al., 2010), but there has not yet been any comprehensive study. Recently, NLP has seen several uses of heterogeneous networks (though not by that name) for use with label propagation algorithms (Das and Petrov, 2011; Spieros et al., 2011) and random walks (Toutanova et al., 2004; Kok and Brockett, 2010).

Several authors have proposed the idea of using network centrality measures to rank the impacts of journals, authors, papers, etc. (Bollen et al., 2006; Bergstrom et al., 2008; Chen et al., 2007; Liu et al., 2005), and it has even been proposed that centrality can be applicable in bipartite networks (Zhou et al., 2007). We propose that Pagerank on any general heterogeneous network is appropriate for creating ranked lists for each type of entity. Most previous papers also lack a robust evaluation, demonstrating agreement with previous methods or with some external awards or recognitions. We use a random graph that replicates the properties of the real-world network to show that Pagerank on the heterogeneous network outperforms other methods.

Name disambiguation has been studied in a number of different settings, including graph-based settings. It is common to use the coauthorship graph (Kang et al., 2009; Fan et al., 2011), but authors

have also used lexical similarity graphs (On and Lee, 2007), citation graphs (McRae-Spencer and Shadbolt, 2006), or social networks (Malin, 2005). Almost all graph methods are unsupervised.

There have been some topic models developed specifically for relational data (Wang et al., 2006; Airoldi et al., 2008), but both of these models have limitations in the types of relational data they are able to model. The group topic model described in (Wang et al., 2006) is able to create stronger topics by considering associations between words, events, and entities, but is very coarse in the way it handles the behavior of entities, and does not generalize to multiple different types of entities. The stochastic blockmodel of (Airoldi et al., 2008) can create blocks of similar entities in a graph and is general in the types of graphs it can handle, but produces less meaningful results on graphs that have specific schemas.

6 Conclusion and Future Directions

In this paper, we present a heterogeneous network treatment of the ACL Anthology Network and demonstrate several applications of it. Using only off-the-shelf graph algorithms with a single data representation, the heterogeneous AAN, we are able to very easily build a scientific impact measure that is more accurate than existing measures, an author disambiguation system better than existing graph-based author disambiguation systems, and a random-walk-based topic model that is competitive with statistical topic models.

While there are many other tasks, such as citation-based summarization, that could likely be approached using this framework with the appropriate addition of new types of nodes into the heterogeneous AAN network, there are even some potential synergies between the tasks described in this paper that have yet to be explored. For example, we may consider that the methods of the author disambiguation or topic modeling tasks could be to find the highest-impact papers associated with a term (for survey generation, perhaps) or high-impact authors associated with a workshop’s topic (to select good reviewers for it). We believe that heterogeneous graphs are a flexible framework that will allow re-

searchers to find simple, flexible solutions for a variety of problems.

Acknowledgments

This research is supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center (DoI/NBC) contract number D11PC20153. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

References

- Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing. 2008. Mixed membership stochastic blockmodels. *The Journal of Machine Learning Research*, 9:1981–2014.
- Réka Albert and Albert-László Barabási. 2002. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47.
- A.L. Barabási and R. Albert. 1999. Emergence of scaling in random networks. *Science*, 286(5439):509–512.
- Carl T. Bergstrom, Jevin D. West, and Marc A. Wiseman. 2008. The eigenfactor metrics. *The Journal of Neuroscience*, 28(45):11433–11434.
- Steven Bird, Robert Dale, Bonnie J Dorr, Bryan Gibson, Mark Joseph, Min-Yen Kan, Dongwon Lee, Brett Powley, Dragomir R Radev, and Yee Fan Tan. 2008. The ACL anthology reference corpus: A reference dataset for bibliographic research in computational linguistics. In *Proc. of the 6th International Conference on Language Resources and Evaluation Conference (LREC08)*, pages 1755–1759.
- D.M. Blei, A.Y. Ng, and M.I. Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.
- Johan Bollen, Marko A. Rodriguez, and Herbert Van de Sompel. 2006. Journal status. *CoRR*, abs/cs/0601030.
- Johan Bollen, Herbert Van de Sompel, Aric Hagberg, and Ryan Chute. 2009. A principal component analysis of 39 scientific impact measures. *PloS one*, 4(6):e6022.
- Tibor Braun, Wolfgang Glänzel, and András Schubert. 2006. A hirsch-type index for journals. *Scientometrics*, 69(1):169–173.
- Jonathan Chang and David M Blei. 2010. Hierarchical relational models for document networks. *The Annals of Applied Statistics*, 4(1):124–150.

- Peng Chen, Huafeng Xie, Sergei Maslov, and Sid Redner. 2007. Finding scientific gems with googles pagerank algorithm. *Journal of Informetrics*, 1(1):8–15.
- Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. 2009. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703.
- Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 600–609.
- Paul Erdős and Alfréd Rényi. 1960. On the evolution of random graphs. *Magyar Tud. Akad. Mat. Kutató Int. Közl.*, 5:17–61.
- Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res. (JAIR)*, 22:457–479.
- Xiaoming Fan, Jianyong Wang, Xu Pu, Lizhu Zhou, and Bing Lv. 2011. On graph-based name disambiguation. *J. Data and Information Quality*, 2(2):10:1–10:23, February.
- William A. Gale, Kenneth W. Church, and David Yarowsky. 1992. Work on statistical methods for word sense disambiguation. In *Working Notes of the AAAI Fall Symposium on Probabilistic Approaches to Natural Language*, volume 54, page 60.
- David Hall, Daniel Jurafsky, and Christopher D. Manning. 2008. Studying the history of ideas using topic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 363–371. ACL.
- Ahmed Hassan, Amjad Abu-Jbara, and Dragomir Radev. 2012. Extracting signed social networks from text. *TextGraphs-7*, page 6.
- Jorge E. Hirsch. 2005. An index to quantify an individual’s scientific research output. *Proceedings of the National Academy of Sciences of the United States of America*, 102(46):16569.
- Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM.
- Kalervo Järvelin and Jaana Kekäläinen. 2000. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 41–48. ACM.
- In-Su Kang, Seung-Hoon Na, Seungwoo Lee, Hanmin Jung, Pyung Kim, Won-Kyung Sung, and Jong-Hyeok Lee. 2009. On co-authorship for author disambiguation. *Information Processing & Management*, 45(1):84–97.
- Brian Karrer and Mark EJ Newman. 2011. Stochastic blockmodels and community structure in networks. *Physical Review E*, 83(1):016107.
- Stanley Kok and Chris Brockett. 2010. Hitting the right paraphrases in good time. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 145–153. ACL.
- Oren Kurland and Lillian Lee. 2005. Pagerank without hyperlinks: Structural reranking using links induced by language models. In *SIGIR ’05*.
- Renaud Lambiotte and Marcel Ausloos. 2006. Collaborative tagging as a tripartite network. *Computational Science–ICCS 2006*, pages 1114–1117.
- David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031.
- R.N. Lichtenwalter, J.T. Lussier, and N.V. Chawla. 2010. New perspectives and methods in link prediction. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 243–252. ACM.
- Xiaoming Liu, Johan Bollen, Michael L. Nelson, and Herbert Van de Sompel. 2005. Co-authorship networks in the digital library research community. *Information processing & management*, 41(6):1462–1480.
- Bo Long, Zhongfei Zhang, and Tianbing Xu. 2008. Clustering on complex graphs. In *Proc. the 23rd Conf. AAAI 2008*.
- Angelica Lozano and Giovanni Storchi. 2002. Shortest viable hyperpath in multimodal networks. *Transportation Research Part B: Methodological*, 36(10):853–874.
- Bradley Malin. 2005. Unsupervised name disambiguation via social network similarity. In *Workshop on Link Analysis, Counterterrorism, and Security*, volume 1401, pages 93–102.
- Sergei Maslov and Sidney Redner. 2008. Promise and pitfalls of extending google’s pagerank algorithm to citation networks. *The Journal of Neuroscience*, 28(44):11103–11105.
- Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530. ACL.
- Duncan M. McRae-Spencer and Nigel R. Shadbolt. 2006. Also by the same author: Aktiveauthor, a citation graph approach to name disambiguation. In *Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries*, pages 53–54. ACM.

- Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In *Proceedings of EMNLP*, volume 4, pages 404–411. Barcelona, Spain.
- Rada Mihalcea. 2005. Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling. In *Proceedings of HLT-EMNLP*, pages 411–418. ACL.
- David Mimno, Hanna M. Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 262–272. ACL.
- Panchanan Mitra. 2006. Hirsch-type indices for ranking institutions scientific research output. *Current Science*, 91(11):1439.
- Tsuyoshi Murata. 2010. Detecting communities from tripartite networks. In *Proceedings of the 19th international conference on World wide web*, pages 1159–1160. ACM.
- Pradeep Muthukrishnan, Dragomir Radev, and Qiaozhu Mei. 2010. Edge weight regularization over multiple graphs for similarity learning. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 374–383. IEEE.
- Mark E.J. Newman and Juyong Park. 2003. Why social networks are different from other types of networks. *Physical Review E*, 68(3):036122.
- Byung-Won On and Dongwon Lee. 2007. Scalable name disambiguation using multi-level graph partition. In *Proceedings of the 7th SIAM international conference on data mining*, pages 575–580.
- Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: bringing order to the web.
- Romualdo Pastor-Satorras and Alessandro Vespignani. 2001. Epidemic spreading in scale-free networks. *Physical review letters*, 86(14):3200–3203.
- Dragomir R. Radev, Pradeep Muthukrishnan, Vahed Qazvinian, and Amjad Abu-Jbara. 2013. The ACL anthology network corpus. *Language Resources and Evaluation*, pages 1–26.
- William M. Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850.
- S. Redner. 1998. How popular is your paper? an empirical study of the citation distribution. *The European Physical Journal B-Condensed Matter and Complex Systems*, 4(2):131–134.
- P. Sarkar, A.W. Moore, and A. Prakash. 2008. Fast incremental proximity search in large graphs. In *Proceedings of the 25th international conference on Machine learning*, pages 896–903. ACM.
- Allan A. Sioson. 2005. *Multimodal networks in biology*. Ph.D. thesis, Virginia Polytechnic Institute and State University.
- Michael Speriosu, Nikita Sudan, Sid Upadhyay, and Jason Baldridge. 2011. Twitter polarity classification with label propagation over lexical links and the follower graph. In *Proceedings of the First workshop on Unsupervised Learning in NLP*, pages 53–63, Edinburgh, Scotland, July. ACL.
- Alexander Strehl and Joydeep Ghosh. 2003. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research*, 3:583–617.
- Yizhou Sun, Jiawei Han, Peixiang Zhao, Zhijun Yin, Hong Cheng, and Tianyi Wu. 2009. Rankclus: integrating clustering with ranking for heterogeneous information network analysis. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pages 565–576. ACM.
- Yizhou Sun, Rick Barber, Manish Gupta, and Jiawei Han. 2011. Co-author relationship prediction in heterogeneous bibliographic networks.
- Yizhou Sun, Charu C. Aggarwal, and Jiawei Han. 2012. Relation strength-aware clustering of heterogeneous information networks with incomplete attributes. *Proceedings of the VLDB Endowment*, 5(5):394–405.
- Ben Taskar, Ming-Fai Wong, Pieter Abbeel, and Daphne Koller. 2003. Link prediction in relational data. In *Neural Information Processing Systems*, volume 15.
- Robert Tibshirani, Guenther Walther, and Trevor Hastie. 2001. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423.
- Kristina Toutanova, Christopher D Manning, and Andrew Y Ng. 2004. Learning random walk models for inducing word dependency distributions. In *Proceedings of the twenty-first international conference on Machine learning*, page 103. ACM.
- Xuerui Wang, Natasha Mohanty, and Andrew McCallum. 2006. Group and topic discovery from relations and their attributes. Technical report, DTIC Document.
- Kai Yu, Shipeng Yu, and Volker Tresp. 2006. Soft clustering on graphs. *Advances in Neural Information Processing Systems*, 18:1553.
- Ding Zhou, Sergey A. Orshanskiy, Hongyuan Zha, and C. Lee Giles. 2007. Co-ranking authors and documents in a heterogeneous network. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 739–744. IEEE.

Discriminative Lexical Semantic Segmentation with Gaps: Running the MWE Gamut

Nathan Schneider Emily Danchik Chris Dyer Noah A. Smith

School of Computer Science

Carnegie Mellon University

Pittsburgh, PA 15213, USA

{nschneid,emilydan,cdyer,nasmith}@cs.cmu.edu

Abstract

We present a novel representation, evaluation measure, and supervised models for the task of identifying the multiword expressions (MWEs) in a sentence, resulting in a *lexical semantic segmentation*. Our approach generalizes a standard chunking representation to encode MWEs containing gaps, thereby enabling efficient sequence tagging algorithms for feature-rich discriminative models. Experiments on a new dataset of English web text offer the first linguistically-driven evaluation of MWE identification with truly heterogeneous expression types. Our statistical sequence model greatly outperforms a lookup-based segmentation procedure, achieving nearly 60% F_1 for MWE identification.

1 Introduction

Language has a knack for defying expectations when put under the microscope. For example, there is the notion—sometimes referred to as *compositionality*—that words will behave in predictable ways, with individual meanings that combine to form complex meanings according to general grammatical principles. Yet language is awash with examples to the contrary: in particular, idiomatic expressions such as *awash with NP*, *have a knack for VP_{ing}*, *to the contrary*, and *defy expectations*. Thanks to processes like metaphor and grammaticalization, these are (to various degrees) semantically opaque, structurally fossilized, and/or statistically idiosyncratic. In other words, idiomatic expressions may be exceptional in form, function, or distribution. They are so diverse, so unruly, so

1. **MW named entities:** *Prime Minister Tony Blair*
2. **MW compounds:** *hot air balloon, skinny dip*
3. **conventionally SW compounds:** *somewhere*
4. **verb-particle:** *pick up, dry out, take over, cut short*
5. **verb-preposition:** *refer to, depend on, look for*
6. **verb-noun(-preposition):** *pay attention (to)*
7. **support verb:** *make decisions, take pictures*
8. **other phrasal verb:** *put up with, get rid of*
9. **PP modifier:** *above board, at all, from time to time*
10. **coordinated phrase:** *cut and dry, more or less*
11. **connective:** *as well as, let alone, in spite of*
12. **semi-fixed VP:** *pick up where <one> left off*
13. **fixed phrase:** *scared to death, leave of absence*
14. **phatic:** *You're welcome. Me neither!*
15. **proverb:** *Beggars can't be choosers.*

Figure 1: Some of the classes of idioms in English. The examples included here contain multiple lexicalized words—with the exception of those in (3), if the conventional single-word (SW) spelling is used.

difficult to circumscribe, that entire theories of syntax are predicated on the notion that constructions with idiosyncratic form-meaning mappings (Fillmore et al., 1988; Goldberg, 1995) or statistical properties (Goldberg, 2006) offer crucial evidence about the grammatical organization of language.

Here we focus on **multiword expressions** (MWEs): *lexicalized* combinations of two or more words that are exceptional enough to be considered as single units in the lexicon. As figure 1 illustrates, MWEs occupy diverse syntactic and semantic functions. Within MWEs, we distinguish (a) proper names and (b) lexical idioms. The latter have proved themselves a “pain in the neck for NLP” (Sag et al., 2002). Automatic and efficient detection of MWEs, though far from solved, would have diverse appli-

cations including machine translation (Carpuat and Diab, 2010), information retrieval (Newman et al., 2012), opinion mining (Berend, 2011), and second language learning (Ellis et al., 2008).

It is difficult to establish any comprehensive taxonomy of multiword idioms, let alone develop linguistic criteria and corpus resources that cut across these types. Consequently, the voluminous literature on MWEs in computational linguistics—see §7, Baldwin and Kim (2010), and Ramisch (2012) for surveys—has been fragmented, looking (for example) at subclasses of phrasal verbs or nominal compounds in isolation. To the extent that MWEs have been annotated in existing corpora, it has usually been as a secondary aspect of some other scheme. Traditionally, such resources have prioritized certain kinds of MWEs to the exclusion of others, so they are not appropriate for evaluating general-purpose identification systems.

In this article, we briefly review a shallow form of analysis for MWEs that is neutral to expression type, and that facilitates free text annotation without requiring a prespecified MWE lexicon (§2). The scheme applies to gappy (discontinuous) as well as contiguous expressions, and allows for a qualitative distinction of association strengths. In Schneider et al. (2014) we have applied this scheme to fully annotate a 55,000-word corpus of English web reviews (Bies et al., 2012a), a conversational genre in which colloquial idioms are highly salient. This article’s main contribution is to show that the representation—constrained according to linguistically motivated assumptions (§3)—can be transformed into a sequence tagging scheme that resembles standard approaches in named entity recognition and other text chunking tasks (§4). Along these lines, we develop a discriminative, structured model of MWEs in context (§5) and train, evaluate, and examine it on the annotated corpus (§6). Finally, in §7 and §8 we comment on related work and future directions.

2 Annotated Corpus

To build and evaluate a multiword expression analyzer, we use the MWE-annotated corpus of Schneider et al. (2014). It consists of informal English web text that has been specifically and completely annotated for MWEs, without reference to any particular

lexicon. To the best of our knowledge, this corpus is the first to be freely annotated for many kinds of MWEs (without reference to a lexicon), and is also the first dataset of social media text with MWE annotations beyond named entities. This section gives a synopsis of the annotation conventions used to develop that resource, as they are important to understanding our models and evaluation.

Rationale. The multiword expressions community has lacked a canonical corpus resource comparable to benchmark datasets used for problems such as NER and parsing. Consequently, the MWE literature has been driven by lexicography: typically, the goal is to acquire an MWE lexicon with little or no supervision, or to apply such a lexicon to corpus data. Studies of MWEs in context have focused on various subclasses of constructions in isolation, necessitating special-purpose datasets and evaluation schemes. By contrast, Schneider et al.’s (2014) corpus creates an opportunity to tackle *general-purpose* MWE identification, such as would be desirable for use by high-coverage downstream NLP systems. It is used to train and evaluate our models below. The corpus is publicly available as a benchmark for further research.¹

Data. The documents in the corpus are online user reviews of restaurants, medical providers, retailers, automotive services, pet care services, etc. Marked by conversational and opinionated language, this genre is fertile ground for colloquial idioms (Nunberg et al., 1994; Moon, 1998). The 723 reviews (55,000 words, 3,800 sentences) in the English Web Treebank (WTB; Bies et al., 2012b) were collected by Google, tokenized, and annotated with phrase structure trees in the style of the Penn Treebank (Marcus et al., 1993). MWE annotators used the sentence and word tokenizations supplied by the treebank.²

Annotation scheme. The annotation scheme itself was designed to be as simple as possible. It consists of grouping together the tokens in each sentence that belong to the same MWE instance. While annotation guidelines provide examples of MWE groupings in a wide range of constructions, the annotator is not

¹<http://www.ark.cs.cmu.edu/LexSem/>

²Because we use treebank data, syntactic parses are available to assist in post hoc analysis. Syntactic information was not shown to annotators.

	# of constituent tokens				# of gaps		
	2	3	≥4	total	0	1	2
<i>strong</i>	2257	595	172	3024	2626	394	4
<i>weak</i>	269	121	69	459	322	135	2
	2526	716	241	3483	2948	529	6

Table 1: Counts in the MWE corpus.

tied to any particular taxonomy or syntactic structure. This simplifies the number of decisions that have to be made for each sentence, even if some are difficult.

Further instructions to annotators included:

- Groups should include only the lexically fixed parts of an expression (modulo inflectional morphology); this generally excludes determiners and pronouns: *made the mistake, pride themselves on*.
- Multiword proper names count as MWEs.
- Misspelled or unconventionally spelled tokens are interpreted according to the intended word if clear.
- Overtokenized words (spelled as two tokens, but conventionally one word) are joined as multiwords. Clitics separated by the tokenization in the corpus—negative *n't*, possessive *'s*, etc.—are joined if functioning as a fixed part of a multiword (e.g., *T 's Cafe*), but not if used productively.

Gaps. There are, broadly speaking, three reasons to group together tokens that are not fully contiguous. Most commonly, gaps contain internal modifiers, such as *good* in *make good decisions*. Syntactic constructions such as the passive can result in gaps that might not otherwise be present: in *good decisions were made*, there is instead a gap filled by the passive auxiliary. Finally, some MWEs may take internal arguments: *they gave me a break*. Figure 1 has additional examples. Multiple gaps can occur even within the same expression, though it is rare: *they agreed to give Bob a well-deserved break*.

Strength. The annotation scheme has two “strength” levels for MWEs. Clearly idiomatic expressions are marked as strong MWEs, while mostly compositional but especially frequent collocations/phrases (e.g., *abundantly clear* and *patently obvious*) are marked as weak MWEs. Weak multiword groups are allowed to include strong MWEs as constituents (but not vice versa). Strong groups are required to cohere when used inside weak groups: that is, a weak group cannot include only part of a strong group. For purposes of annotation, there were no constraints

hinging on the ordering of tokens in the sentence.

Process. MWE annotation proceeded one sentence at a time. The 6 annotators referred to and improved the guidelines document on an ongoing basis. Every sentence was seen independently by at least 2 annotators, and differences of opinion were discussed and resolved (often by marking a weak MWE as a compromise). See Schneider et al. (2014) for details.

Statistics. The annotated corpus consists of 723 documents (3,812 sentences). MWEs are frequent in this domain: 57% of sentences (72% of sentences over 10 words long) and 88% of documents contain at least one MWE. $8,060/55,579=15\%$ of tokens belong to an MWE; in total, there are 3,483 MWE instances. 544 (16%) are strong MWEs containing a gold-tagged proper noun—most are proper names. A breakdown appears in table 1.

3 Representation and Task Definition

We define a **lexical segmentation** of a sentence as a partitioning of its tokens into segments such that each segment represents a single unit of lexical meaning. A *multiword* lexical expression may contain **gaps**, i.e. interruptions by other segments. We impose two restrictions on gaps that appear to be well-motivated linguistically:

- **Projectivity:** Every expression filling a gap must be completely contained within that gap; gappy expressions may not interleave.
- **No nested gaps:** A gap in an expression may be filled by other single- or multiword expressions, so long as those do not themselves contain gaps.

Formal grammar. Our scheme corresponds to the following extended CFG (Thatcher, 1967), where S is the full sentence and terminals w are word tokens:

$$\begin{aligned} S &\rightarrow X^+ \\ X &\rightarrow \underline{w}^+ (Y^+ \underline{w}^+)^* \\ Y &\rightarrow \underline{w}^+ \end{aligned}$$

Each expression X or Y is lexicalized by the words in one or more underlined variables on the right-hand side. An X constituent may optionally contain one or more gaps filled by Y constituents, which must not contain gaps themselves.³

³MWEs with multiple gaps are rare but attested in data: e.g., *putting me at my ease*. We encountered one violation of the gap nesting constraint in the reviews data: *I have₁² nothing₁² but₁² fantastic things₂² to₁² say₁²*. Additionally, the interrupted phrase

Denoting multiword groupings with subscripts, *My wife had taken₁ her '07₂ Ford₂ Fusion₂ in₁ for a routine oil₃ change₃* contains 3 multiword groups— $\{taken, in\}$, $\{'07, Ford, Fusion\}$, $\{oil, change\}$ —and 7 single-word groups. The first MWE is gappy (accentuated by the box); a single word and a contiguous multiword group fall within the gap. The projectivity constraint forbids an analysis like *taken₁ her '07₂ Ford₁ Fusion₂*, while the gap nesting constraint forbids *taken₁ her₂ '07 Ford₂ Fusion₂ in₁*.

3.1 Two-level Scheme: Strong vs. Weak MWEs

Our annotated data distinguish two strengths of MWEs as discussed in §2. Augmenting the grammar of the previous section, we therefore designate nonterminals as strong (\bar{X} , \bar{Y}) or weak (\tilde{X} , \tilde{Y}):

$$\begin{aligned} S &\rightarrow \tilde{X}^+ \\ \tilde{X} &\rightarrow \bar{X}^+ (\tilde{Y}^+ \bar{X}^+)^* \\ \bar{X} &\rightarrow \underline{w}^+ (\tilde{Y}^+ \underline{w}^+)^* \\ \tilde{Y} &\rightarrow \underline{y}^+ \\ \bar{Y} &\rightarrow \underline{w}^+ \end{aligned}$$

A weak MWE may be lexicalized by single words and/or strong multiwords. Strong multiwords cannot contain weak multiwords except in gaps. Further, the contents of a gap cannot be part of any multiword that extends outside the gap.⁴

For example, consider the segmentation: *he was willing to budge₁ a₂ little₂ on₁ the price which means⁴ a₃⁴ lot₃⁴ to⁴ me⁴*. Subscripts denote strong MW groups and superscripts weak MW groups; unmarked tokens serve as single-word expressions. The MW groups are thus $\{budge, on\}$, $\{a, little\}$, $\{a, lot\}$, and $\{means, \{a, lot\}, to, me\}$. As should be evident from the grammar, the projectivity and gap-nesting constraints apply here just as in the 1-level scheme.

3.2 Evaluation

Matching criteria. Given that most tokens do not belong to an MWE, to evaluate MWE identification we adopt a precision/recall-based measure from the coreference resolution literature. The MUC criterion (Vilain et al., 1995) measures precision and recall

great gateways never¹ before¹, so₃² far₃² as₃² Hudson knew², seen¹ by Europeans was annotated in another corpus.

⁴This was violated 6 times in our annotated data: modifiers within gaps are sometimes collocated with the gappy expression, as in *on₂¹ a₂¹ tight¹ budget₂¹ and have₂¹ little¹ doubt₂¹*.

of links in terms of groups (units) implied by the transitive closure over those links.⁵ It can be defined as follows:

Let $a - b$ denote a link between two elements in the gold standard, and $a \hat{=} b$ denote a link in the system prediction. Let the $*$ operator denote the transitive closure over all links, such that $\llbracket a -^* b \rrbracket$ is 1 if a and b belong to the same (gold) set, and 0 otherwise. Assuming there are no redundant⁶ links within any annotation (which in our case is guaranteed by linking consecutive words in each MWE), we can write the MUC precision and recall measures as:

$$P = \frac{\sum_{a,b:a \hat{=} b} \llbracket a -^* b \rrbracket}{\sum_{a,b:a \hat{=} b} 1} \quad R = \frac{\sum_{a,b:a-b} \llbracket a \hat{=}^* b \rrbracket}{\sum_{a,b:a-b} 1}$$

This awards partial credit when predicted and gold expressions overlap in part. Requiring full MWEs to match exactly would arguably be too stringent, overpenalizing larger MWEs for minor disagreements. We combine precision and recall using the standard F_1 measure of their harmonic mean. This is the **link-based** evaluation used for most of our experiments. For comparison, we also report some results with a more stringent **exact match** evaluation where the span of the predicted MWE must be identical to the span of the gold MWE for it to count as correct.

Strength averaging. Recall that the 2-level scheme (§3.1) distinguishes *strong* vs. *weak* links/groups, where the latter category applies to reasonably compositional collocations as well as ambiguous or difficult cases. If where one annotation uses a weak link the other has a strong link or no link at all, we want to penalize the disagreement less than if one had a strong link and the other had no link. To accommodate the 2-level scheme, we therefore average F_1^\uparrow , in which all weak links have been converted to strong links, and F_1^\downarrow , in which they have been removed: $F_1 = \frac{1}{2}(F_1^\uparrow + F_1^\downarrow)$.⁷ If neither annotation contains any weak links, this equals the MUC

⁵As a criterion for coreference resolution, the MUC measure has perceived shortcomings which have prompted several other measures (see Recasens and Hovy, 2011 for a review). It is not clear, however, whether any of these criticisms are relevant to MWE identification.

⁶A link between a and b is redundant if the other links already imply that a and b belong to the same set. A set of N elements is expressed non-redundantly with exactly $N - 1$ links.

⁷Overall precision and recall are likewise computed by averaging “strengthened” and “weakened” measurements.

no gaps,	he was willing to budge $\overbrace{\text{a little on the price which means a lot to me}}$.										$(0 BI^+)^+$						
1-level	0	0	0	0	0	B	I	0	0	0	0	B	I	I	I	I	0
no gaps,	he was willing to budge $\overbrace{\text{a little on the price which means a lot to me}}$.										$(0 B[\tilde{I}\tilde{I}]^+)^+$						
2-level	0	0	0	0	0	B	\tilde{I}	0	0	0	0	B	\tilde{I}	\tilde{I}	\tilde{I}	\tilde{I}	0
gappy,	he was willing to budge $\overbrace{\text{a little on the price which means a lot to me}}$.										$(0 B(o bi^+ I)^*I^+)^+$						
1-level	0	0	0	0	B	b	i	I	0	0	0	B	I	I	I	I	0
gappy,	he was willing to budge $\overbrace{\text{a little on the price which means a lot to me}}$.										$(0 B(o b[\tilde{i}\tilde{i}]^+ [\tilde{I}\tilde{I}]^*)^*[\tilde{I}\tilde{I}]^+)^+$						
2-level	0	0	0	0	B	b	\tilde{i}	\tilde{I}	0	0	0	B	\tilde{I}	\tilde{I}	\tilde{I}	\tilde{I}	0

Figure 2: Examples and regular expressions for the 4 tagging schemes. Strong links are depicted with solid arcs, and weak links with dotted arcs. The bottom analysis was provided by an annotator; the ones above are simplifications.

score because $F_1 = F_1^\uparrow = F_1^\downarrow$. This method applies to both the link-based and exact match evaluation criteria.

4 Tagging Schemes

Following (Ramshaw and Marcus, 1995), shallow analysis is often modeled as a sequence-chunking task, with tags containing chunk-positional information. The BIO scheme and variants (e.g., BILOU; Ratnoff and Roth, 2009) are standard for tasks like named entity recognition, supersense tagging, and shallow parsing.

The language of derivations licensed by the grammars in §3 allows for a tag-based encoding of MWE analyses with only bigram constraints. We describe 4 tagging schemes for MWE identification, starting with BIO and working up to more expressive variants. They are depicted in figure 2.

No gaps, 1-level (3 tags). This is the standard contiguous chunking representation from Ramshaw and Marcus (1995) using the tags $\{0 B I\}$. 0 is for tokens **outside** any chunk; B marks tokens **beginning** a chunk; and I marks other tokens **inside** a chunk. Multiword chunks will thus start with B and then I. B must always be followed by I; I is not allowed at the beginning of the sentence or following 0.

No gaps, 2-level (4 tags). We can distinguish strength levels by splitting I into two tags: \tilde{I} for strong expressions and \tilde{I} for weak expressions. To express strong and weak contiguous chunks requires 4 tags: $\{0 B \tilde{I} \tilde{I}\}$. (Marking B with a strength as well would be redundant because MWEs are never length-one chunks.) The constraints on \tilde{I} and \tilde{I} are the same as the constraints on I in previous schemes. If \tilde{I} and

\tilde{I} occur next to each other, the strong attachment will receive higher precedence, resulting in analysis of strong MWEs as nested within weak MWEs.

Gappy, 1-level (6 tags). Because gaps cannot themselves contain gappy expressions (we do not support full recursivity), a finite number of additional tags are sufficient to encode gappy chunks. We therefore add lowercase tag variants representing tokens *within a gap*: $\{0 o B b I i\}$. In addition to the constraints stated above, no within-gap tag may occur at the beginning or end of the sentence or immediately following or preceding 0. Within a gap, b, i, and o behave like their out-of-gap counterparts.

Gappy, 2-level (8 tags). 8 tags are required to encode the 2-level scheme with gaps: $\{0 o B b \tilde{I} \tilde{i} \tilde{I} \tilde{i}\}$. Variants of the inside tag are marked for strength of the incoming link—this applies gap-externally (capitalized tags) and gap-internally (lowercase tags). If \tilde{I} or \tilde{i} immediately follows a gap, its diacritic reflects the strength of the gappy expression, not the gap’s contents.

5 Model

With the above representations we model MWE identification as sequence tagging, one of the paradigms that has been used previously for identifying *contiguous* MWEs (Constant and Sigogne, 2011, see §7).⁸ Constraints on legal tag bigrams are sufficient to ensure the full tagging is well-formed subject to the regular expressions in figure 2; we enforce these

⁸Hierarchical modeling based on our representations is left to future work.

constraints in our experiments.⁹

In NLP, conditional random fields (Lafferty et al., 2001) and the structured perceptron (Collins, 2002) are popular techniques for discriminative sequence modeling with a convex loss function. We choose the second approach for its speed: learning and inference depend mainly on the runtime of the Viterbi algorithm, whose asymptotic complexity is linear in the length of the input and (with a first-order Markov assumption) quadratic in the number of tags. Below, we review the structured perceptron and discuss our cost function, features, and experimental setup.

5.1 Cost-Augmented Structured Perceptron

The structured perceptron’s (Collins, 2002) learning procedure, algorithm 1, generalizes the classic perceptron algorithm (Freund and Schapire, 1999) to incorporate a structured decoding step (for sequences, the Viterbi algorithm) in the inner loop. Thus, training requires only max inference, which is fast with a first-order Markov assumption. In training, features are adjusted where a tagging error is made; the procedure can be viewed as optimizing the structured hinge loss. The output of learning is a weight vector that parametrizes a feature-rich scoring function over candidate labelings of a sequence.

To better align the learning algorithm with our F -score-based MWE evaluation (§3.2), we use a cost-augmented version of the structured perceptron that is sensitive to different kinds of errors during training. When recall is the bigger obstacle, we can adopt the following cost function: given a sentence \mathbf{x} , its gold labeling \mathbf{y}^* , and a candidate labeling \mathbf{y}' ,

$$\text{cost}(\mathbf{y}^*, \mathbf{y}', \mathbf{x}) = \sum_{j=1}^{|\mathbf{y}^*|} c(y_j^*, y_j') \quad \text{where}$$

$$c(y^*, y') = \llbracket y^* \neq y' \rrbracket + \rho \llbracket y^* \in \{\mathbf{B}, \mathbf{b}\} \wedge y' \in \{\mathbf{0}, \mathbf{o}\} \rrbracket$$

A single nonnegative hyperparameter, ρ , controls the tradeoff between recall and accuracy; higher ρ biases the model in favor of recall (possibly hurting accuracy and precision). This is a slight variant of the recall-oriented cost function of Mohit et al. (2012). The difference is that we only penalize *beginning-of-expression* recall errors. Preliminary

⁹The 8-tag scheme licenses 42 tag bigrams: sequences such as $\mathbf{B} \mathbf{0}$ and $\mathbf{o} \bar{\mathbf{i}}$ are prohibited. There are also constraints on the allowed tags at the beginning and end of the sequence.

```

Input: data  $\langle (\mathbf{x}^{(n)}, \mathbf{y}^{(n)}) \rangle_{n=1}^N$ ; number of iterations  $M$ 
 $\mathbf{w} \leftarrow \mathbf{0}$ 
 $\bar{\mathbf{w}} \leftarrow \mathbf{0}$ 
 $t \leftarrow 1$ 
for  $m = 1$  to  $M$  do
  for  $n = 1$  to  $N$  do
     $\langle \mathbf{x}, \mathbf{y} \rangle \leftarrow \langle \mathbf{x}^{(n)}, \mathbf{y}^{(n)} \rangle$ 
     $\hat{\mathbf{y}} \leftarrow \arg \max_{\mathbf{y}'} (\mathbf{w}^\top \mathbf{g}(\mathbf{x}, \mathbf{y}') + \text{cost}(\mathbf{y}, \mathbf{y}', \mathbf{x}))$ 
    if  $\hat{\mathbf{y}} \neq \mathbf{y}$  then
       $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{g}(\mathbf{x}, \mathbf{y}) - \mathbf{g}(\mathbf{x}, \hat{\mathbf{y}})$ 
       $\bar{\mathbf{w}} \leftarrow \bar{\mathbf{w}} + t \mathbf{g}(\mathbf{x}, \mathbf{y}) - t \mathbf{g}(\mathbf{x}, \hat{\mathbf{y}})$ 
    end
     $t \leftarrow t + 1$ 
  end
end
Output:  $\mathbf{w} - (\bar{\mathbf{w}}/t)$ 

```

Algorithm 1: Training with the averaged perceptron. (Adapted from Daumé, 2006, p. 19.)

experiments showed that a cost function penalizing all recall errors—i.e., with $\rho \llbracket \mathbf{y}^* \neq \mathbf{0} \wedge \mathbf{y}' = \mathbf{0} \rrbracket$ as the second term, as in Mohit et al.—tended to append additional tokens to high-confidence MWEs (such as proper names) rather than encourage new MWEs, which would require positing at least two new non-outside tags.

5.2 Features

Basic features. These are largely based on those of Constant et al. (2012): they look at word unigrams and bigrams, character prefixes and suffixes, and POS tags, as well as lexicon entries that match lemmas¹⁰ of multiple words in the sentence. Appendix A lists the basic features in detail.

Some of the basic features make use of *lexicons*. We use or construct 10 lists of English MWEs: all multiword entries in **WordNet** (Fellbaum, 1998); all multiword chunks in **SemCor** (Miller et al., 1993); all multiword entries in English **Wiktionary**;¹¹ the **WikiMwe** dataset mined from English Wikipedia (Hartmann et al., 2012); the **SAID** database of phrasal lexical idioms (Kuiper et al., 2003); the named entities and other MWEs in the WSJ corpus on the English side of the **CEDT** (Hajič et al., 2012);

¹⁰The WordNet API in NLTK (Bird et al., 2009) was used for lemmatization.

¹¹<http://en.wiktionary.org>; data obtained from <https://toolserver.org/~enwikt/definitions/enwikt-defs-20130814-en.tsv.gz>

	entries	LOOKUP					SUPERVISED MODEL			
		max gap length	\bar{P}	\bar{R}	\bar{F}_1	σ	\bar{P}	\bar{R}	\bar{F}_1	σ
<i>preexisting lexicons</i>										
none	0						74.39	44.43	55.57	2.19
WordNet + SemCor	71k	0	<u>46.15</u>	28.41	35.10	2.44	74.51	45.79	56.64	1.90
6 lexicons	420k	0	35.05	46.76	<u>40.00</u>	2.88	76.08	52.39	61.95	1.67
10 lexicons	437k	0	33.98	<u>47.29</u>	39.48	2.88	75.95	51.39	61.17	2.30
best configuration with in-domain lexicon		1	46.66	47.90	47.18	2.31	76.64	51.91	61.84	1.65
			2 lexicons + $MWtypes(train)_{\geq 1}$				6 lexicons + $MWtypes(train)_{\geq 2}$			

Table 2: Use of lexicons for lookup-based vs. statistical segmentation. Supervised learning used only basic features and the structured perceptron, with the 8-tag scheme. Results are with the link-based matching criterion for evaluation.

Top: Comparison of preexisting lexicons. “6 lexicons” refers to WordNet and SemCor plus SAID, WikiMwe, Phrases.net, and English Wiktionary; “10 lexicons” adds MWEs from CEDT, VNC, LVC, and Oyz. (In these lookup-based configurations, allowing gappy MWEs never helps performance.)

Bottom: Combining preexisting lexicons with a lexicon derived from MWEs annotated in the training portion of each cross-validation fold at least once (lookup) or twice (model).

All precision, recall, and F_1 percentages are averaged across 8 folds of cross-validation on **train**; standard deviations are shown for the F_1 score. In each column, the highest value using only preexisting lexicons is underlined, and the highest overall value is bolded. The boxed row indicates the configuration used as the basis for subsequent experiments.

the **verb-particle constructions** (VPCs) dataset of (Baldwin, 2008); a list of **light verb constructions** (LVCs) provided by Claire Bonial; and two idioms websites.¹² After preprocessing, each lexical entry consists of an ordered sequence of word lemmas, some of which may be variables like *<something>*.

Given a sentence and one or more of the lexicons, lookup proceeds as follows: we enumerate entries whose lemma sequences match a sequence of lemmatized tokens, and build a lattice of possible analyses over the sentence. We find the shortest path (i.e., using as few expressions as possible) with dynamic programming, allowing gaps of up to length 2.¹³

Unsupervised word clusters. Distributional clustering on large (unlabeled) corpora can produce lexical generalizations that are useful for syntactic and semantic analysis tasks (e.g.: Miller et al., 2004; Koo et al., 2008; Turian et al., 2010; Owoputi et al., 2013; Grave et al., 2013). We were interested to see whether a similar pattern would hold for MWE identification, given that MWEs are concerned with what is lexically *idiosyncratic*—i.e., backing off from specific lexemes to word classes may lose the MWE-relevant information. Brown clustering¹⁴ (Brown et al., 1992)

¹²<http://www.phrases.net/> and <http://home.postech.ac.kr/~oyz/doc/idiom.html>

¹³Each top-level lexical expression (single- or multiword) incurs a cost of 1; each expression within a gap has cost 1.25.

¹⁴With Liang’s (2005) implementation: <https://github.com/percyliang/brown-cluster>. We obtain 1,000 clusters

on the 21-million-word Yelp Academic Dataset¹⁵ (which is similar in genre to the annotated web reviews data) gives us a hard clustering of word types. To our tagger, we add features mapping the previous, current, and next token to Brown cluster IDs. The feature for the current token conjoins the word lemma with the cluster ID.

Part-of-speech tags. We compared three PTB-style POS taggers on the full REVIEWS subcorpus (**train+test**). The Stanford CoreNLP tagger¹⁶ (Toutanova et al., 2003) yields an accuracy of 90.4%. The ARK TweetNLP tagger v. 0.3.2 (Owoputi et al., 2013) achieves 90.1% with the model¹⁷ trained on the Twitter corpus of Ritter et al. (2011), and 94.9% when trained on the ANSWERS, EMAIL, NEWSGROUP, and WEBLOG subcorpora of WTB. We use this third configuration to produce automatic POS tags for training and testing our MWE tagger. (A comparison condition in §6.3 uses oracle POS tags.)

5.3 Experimental Setup

The corpus of web reviews described in §2 is used for training and evaluation. 101 arbitrarily chosen documents (500 sentences, 7,171 words) were held

from words appearing at least 25 times.

¹⁵https://www.yelp.com/academic_dataset

¹⁶v. 3.2.0, with english-bidirectional-distsim

¹⁷http://www.ark.cs.cmu.edu/TweetNLP/model.ritter_ptb_alldata_fixed.20130723

configuration	M	ρ	w	LINK-BASED			EXACT MATCH		
				P	R	F_1	P	R	F_1
base model	5	—	1,765k	69.27	50.49	58.35	60.99	48.27	53.85
+ recall cost	4	150	1,765k	61.09	57.94	59.41	53.09	55.38	54.17
+ clusters	3	100	2,146k	63.98	55.51	59.39	56.34	53.24	54.70
+ oracle POS	4	100	2,145k	66.19	59.35	62.53	58.51	57.00	57.71

Table 3: Comparison of supervised models on **test** (using the 8-tag scheme). The base model corresponds to the boxed result in table 2, but here evaluated on **test**. For each configuration, the number of training iterations M and (except for the base model) the recall-oriented hyperparameter ρ were tuned by cross-validation on **train**.

out as a final **test** set. This left 3,312 sentences/48,408 words for training/development (**train**). Feature engineering and hyperparameter tuning were conducted with 8-fold cross-validation on **train**. The 8-tag scheme is used except where otherwise noted.

In learning with the structured perceptron (algorithm 1), we employ two well-known techniques that can both be viewed as regularization. First, we use the average of parameters over all timesteps of learning. Second, within each cross-validation fold, we determine the number of training iterations (epochs) M by early stopping—that is, after each iteration, we use the model to decode the held-out data, and when that accuracy ceases to improve, use the previous model. The two hyperparameters are the number of iterations and the value of the recall cost hyperparameter (ρ). Both are tuned via cross-validation on **train**; we use the multiple of 50 that maximizes average link-based F_1 . The chosen values are shown in table 3. Experiments were managed with the ducttape tool.¹⁸

6 Results

We experimentally address the following questions to probe and justify our modeling approach.

6.1 Is supervised learning necessary?

Previous MWE identification studies have found benefit to statistical learning over heuristic lexicon lookup (Constant and Sigogne, 2011; Green et al., 2012). Our first experiment tests whether this holds for comprehensive MWE identification: it compares our supervised tagging approach with baselines of heuristic lookup on preexisting lexicons. The baselines construct a lattice for each sentence using the same method as lexicon-based model features (§5.2). If multiple lexicons are used, the union of their en-

tries is used to construct the lattice. The resulting segmentation—which does not encode a strength distinction—is evaluated against the gold standard.

Table 2 shows the results. Even with just the labeled training set as input, the supervised approach beats the strongest heuristic baseline (that incorporates in-domain lexicon entries extracted from the training data) by 30 precision points, while achieving comparable recall. For example, the baseline (but not the statistical model) incorrectly predicts an MWE in *places to eat in Baltimore* (because *eat in*, meaning ‘eat at home,’ is listed in WordNet). The supervised approach has learned not to trust WordNet too much due to this sort of ambiguity. Downstream applications that currently use lexicon matching for MWE identification (e.g., Ghoneim and Diab, 2013) likely stand to benefit from our statistical approach.

6.2 How best to exploit MWE lexicons (type-level information)?

For statistical tagging (right portion of table 2), using more *preexisting* (out-of-domain) lexicons generally improves recall; precision also improves a bit.

A lexicon of MWEs occurring in the non-held-out training data *at least twice*¹⁹ (table 2, bottom right) is marginally worse (better precision/worse recall) than the best result using only preexisting lexicons.

6.3 Variations on the base model

We experiment with some of the modeling alternatives discussed in §5. Results appear in table 3 under both the link-based and exact match evaluation criteria. We note that the exact match scores are (as expected) several points lower.

¹⁹If we train with access to the full lexicon of *training set* MWEs, the learner credulously overfits to relying on that lexicon—after all, it has perfect coverage of the training data!—which proves fatal for the model at test time.

¹⁸<https://github.com/jhclark/ducttape/>

Recall-oriented cost. The recall-oriented cost adds about 1 link-based F_1 point, sacrificing precision in favor of recall.

Unsupervised word clusters. When combined with the recall-oriented cost, these produce a slight improvement to precision/degradation to recall, improving exact match F_1 but not affecting link-based F_1 . Only a few clusters receive high positive weight; one of these consists of *matter, joke, biggie, pun, avail, clue, corkage, frills, worries*, etc. These words are diverse semantically, but all occur in collocations with *no*, which is what makes the cluster coherent and useful to the MWE model.

Oracle part-of-speech tags. Using human-annotated rather than automatic POS tags improves MWE identification by about 3 F_1 points on **test** (similar differences were observed in development).

6.4 What are the highest-weighted features?

An advantage of the linear modeling framework is that we can examine learned feature weights to gain some insight into the model’s behavior.

In general, the highest-weighted features are the lexicon matching features and features indicative of proper names (POS tag of proper noun, capitalized word not at the beginning of the sentence, etc.).

Despite the occasional cluster capturing collocational or idiomatic groupings, as described in the previous section, the clusters appear to be mostly useful for identifying words that tend to belong (or not) to proper names. For example, the cluster with *street, road, freeway, highway, airport*, etc., as well as words outside of the cluster vocabulary, weigh in favor of an MWE. A cluster with everyday destinations (*neighborhood, doctor, hotel, bank, dentist*) prefers non-MWEs, presumably because these words are not typically part of proper names in this corpus. This was from the best model using non-oracle POS tags, so the clusters are perhaps useful in correcting for proper nouns that were mistakenly tagged as common nouns. One caveat, though, is that it is hard to discern the impact of these specific features where others may be capturing essentially the same information.

6.5 How heterogeneous are learned MWEs?

On **test**, the final model (with automatic POS tags) predicts 365 MWE instances (31 are gappy; 23 are

POS pattern	# examples (lowercased lemmas)
NOUN NOUN	53 <i>customer service, oil change</i>
VERB PREP	36 <i>work with, deal with, yell at</i>
PROPN PROP	29 <i>eagle transmission, comfort zone</i>
ADJ NOUN	21 <i>major award, top notch, mental health</i>
VERB PART	20 <i>move out, end up, pick up, pass up</i>
VERB ADV	17 <i>come back, come in, come by, stay away</i>
PREP NOUN	12 <i>on time, in fact, in cash, for instance</i>
VERB NOUN	10 <i>take care, make money, give crap</i>
VERB PRON	10 <i>thank you, get it</i>
PREP PREP	8 <i>out of, due to, out ta, in between</i>
ADV ADV	6 <i>no matter, up front, at all, early on</i>
DET NOUN	6 <i>a lot, a little, a bit, a deal</i>
VERB DET NOUN	6 <i>answer the phone, take a chance</i>
NOUN PREP	5 <i>kind of, care for, tip on, answer to</i>

Table 4: Top predicted POS patterns and frequencies.

weak). There are 298 unique MWE types.

Organizing the predicted MWEs by their coarse POS sequence reveals that the model is not too prejudiced in the kinds of expressions it recognizes: the 298 types fall under 89 unique POS+strength patterns. Table 4 shows the 14 POS sequences predicted 5 or more times as strong MWEs. Some of the examples (*major award, a deal, tip on*) are false positives, but most are correct. Singleton patterns include PROP (worth it), and PREP VERB PREP (*to die for*).

True positive MWEs mostly consist of (a) named entities, and (b) lexical idioms seen in training and/or listed in one of the lexicons. Occasionally the system correctly guesses an unseen and OOV idiom based on features such as hyphenation (*walk - in*) and capitalization/OOV words (*Chili Relleno, BIG MISTAKE*). On **test**, 244 gold MWE types were unseen in training; the system found 93 true positives (where the type was predicted at least once), 109 false positives, and 151 false negatives—an unseen type recall rate of 38%. Removing types that occurred in lexicons leaves 35 true positives, 61 false positives, and 111 false negatives—a unseen and OOV type recall rate of 24%.

6.6 What kinds of mismatches occur?

Inspection of the output turns up false positives due to ambiguity (e.g., *Spongy and sweet bread*); false negatives (*top to bottom*); and overlap (*get high quality service*, gold *get high quality service*; *live up to*, gold *live up to*). A number of the mismatches turn

scheme	$ \mathcal{Y} $	ρ	\bar{M}	$ \bar{\mathbf{w}} $	\bar{P}	\bar{R}	\bar{F}_1
no gaps, 1-level	3	100	2.1	733k	73.33	55.72	63.20
no gaps, 2-level	4	150	3.3	977k	72.60	59.11	65.09
gappy, 1-level	6	200	1.6	1,466k	66.48	61.26	63.65
gappy, 2-level	8	100	3.5	1,954k	73.27	60.44	66.15

Table 5: Training with different tagging schemes. Results are cross-validation averages on **train**. All schemes are evaluated against the full gold standard (8 tags).

out to be problems with the gold standard, like *having our water shut off* (gold *having our water shut off*). This suggests that even noisy automatic taggers might help identify annotation inconsistencies and errors for manual correction.

6.7 Are gappiness and the strength distinction learned in practice?

Three quarters of MWEs are strong and contain no gaps. To see whether our model is actually sensitive to the phenomena of gappiness and strength, we train on data simplified to remove one or both distinctions—as in the first 3 labelings in figure 2—and evaluate against the full 8-tag scheme. For the model with the recall cost, clusters, and oracle POS tags, we evaluate each of these simplifications of the training data in table 5. The gold standard for evaluation remains the same across all conditions.

If the model was unable to recover gappy expressions or the strong/weak distinction, we would expect it to do no better when trained with the full tagset than with the simplified tagset. However, there is some loss in performance as the tagset for learning is simplified, which suggests that gappiness and strength are being learned to an extent.

7 Related Work

Our annotated corpus (Schneider et al., 2014) joins several resources that indicate certain varieties of MWEs: lexicons such as WordNet (Fellbaum, 1998), SAID (Kuiper et al., 2003), and WikiMwe (Hartmann et al., 2012); targeted lists (Baldwin, 2005, 2008; Cook et al., 2008; Tu and Roth, 2011, 2012); web-sites like Wiktionary and Phrases.net; and large-scale corpora such as SemCor (Miller et al., 1993), the French Treebank (Abeillé et al., 2003), the Szeged-ParalellFX corpus (Vincze, 2012), and the Prague Czech-English Dependency Treebank (Čmejrek et al.,

2005). The difference is that Schneider et al. (2014) pursued a *comprehensive annotation approach* rather than targeting specific varieties of MWEs or relying on a preexisting lexical resource. The annotations are *shallow*, not relying explicitly on syntax (though in principle they could be mapped onto the parses in the Web Treebank).

In terms of modeling, the use of machine learning classification (Hashimoto and Kawahara, 2008; Shigeto et al., 2013) and specifically BIO sequence tagging (Diab and Bhutada, 2009; Constant and Sigogne, 2011; Constant et al., 2012; Vincze et al., 2013) for contextual recognition of MWEs is not new. Lexical semantic classification tasks like named entity recognition (e.g., Ratinov and Roth, 2009), sense tagging (Ciaramita and Altun, 2006; Paaß and Reichartz, 2009), and index term identification (Newman et al., 2012) also involve chunking of certain MWEs. But our discriminative models, facilitated by the new corpus, *broaden the scope of the MWE identification task* to include many varieties of MWEs at once, including explicit marking of gaps and a strength distinction. By contrast, the aforementioned identification systems, as well as some MWE-enhanced syntactic parsers (e.g., Green et al., 2012), have been restricted to contiguous MWEs. However, Green et al. (2011) allow gaps to be described as constituents in a syntax tree. Gimpel and Smith’s (2011) shallow, gappy language model allows arbitrary token groupings within a sentence, whereas our model imposes projectivity and nesting constraints (§3). Blunsom and Baldwin (2006) present a sequence model for HPSG supertagging, and evaluate performance on discontinuous MWEs, though the sequence model treats the non-adjacent component supertags like other labels—it cannot enforce that they mutually require one another, as we do via the gappy tagging scheme (§3.1). The lexicon lookup procedures of Bejček et al. (2013) can match gappy MWEs, but are nonstatistical and extremely error-prone when tuned for high oracle recall.

Another major thread of research has pursued *unsupervised* discovery of multiword types from raw corpora, such as with statistical association measures (Church et al., 1991; Pecina, 2010; Ramisch et al., 2012, *inter alia*), parallel corpora (Melamed, 1997; Moirón and Tiedemann, 2006; Tsvetkov and Wintner, 2010), or a combination thereof (Tsvetkov and

Wintner, 2011); this may be followed by a lookup-and-classify approach to contextual identification (Ramisch et al., 2010). Though preliminary experiments with our models did not show benefit to incorporating such automatically constructed lexicons, we hope these two perspectives can be brought together in future work.

8 Conclusion

This article has presented the first supervised model for identifying heterogeneous multiword expressions in English text. Our feature-rich discriminative sequence tagger performs shallow chunking with a novel scheme that allows for MWEs containing gaps, and includes a strength distinction to separate highly idiomatic expressions from collocations. It is trained and evaluated on a corpus of English web reviews that are comprehensively annotated for multiword expressions. Beyond the training data, its features incorporate evidence from external resources—several lexicons as well as unsupervised word clusters; we show experimentally that this statistical approach is far superior to identifying MWEs by heuristic lexicon lookup alone. Future extensions might integrate additional features (e.g., exploiting statistical association measures computed over large corpora), enhance the lexical representation (e.g., by adding semantic tags), improve the expressiveness of the model (e.g., with higher-order features and inference), or integrate the model with other tasks (such as parsing and translation).

Our data and open source software are released at <http://www.ark.cs.cmu.edu/LexSem/>.

Acknowledgments

This research was supported in part by NSF CAREER grant IIS-1054319, Google through the Reading is Believing project at CMU, and DARPA grant FA8750-12-2-0342 funded under the DEFT program. We are grateful to Kevin Knight, Martha Palmer, Claire Bonial, Lori Levin, Ed Hovy, Tim Baldwin, Omri Abend, members of JHU CLSP, the NLP group at Berkeley, and the Noah’s ARK group at CMU, and anonymous reviewers for valuable feedback.

A Basic Features

All are conjoined with the current label, y_i .

Label Features

1. previous label (the only first-order feature)

Token Features

Original token

2. $i = \{1, 2\}$
3. $i = |\mathbf{w}| - \{0, 1\}$
4. capitalized $\wedge \llbracket i = 0 \rrbracket$
5. word shape

Lowercased token

6. prefix: $[w_i]_1^k \Big|_{k=1}^4$
7. suffix: $[w_i]_j^{|w|} \Big|_{j=|w|-3}^{|w|}$
8. has digit
9. has non-alphanumeric c
10. context word: $w_j \Big|_{j=i-2}^{i+2}$
11. context word bigram: $\mathbf{w}_j^{j+1} \Big|_{j=i-2}^{i+1}$

Lemma Features

12. lemma + context lemma if one of them is a verb and the other is a noun, verb, adjective, adverb, preposition, or particle: $\lambda_i \wedge \lambda_j \Big|_{j=i-2}^{i+2}$

Part-of-speech Features

13. context POS: $pos_j \Big|_{j=i-2}^{i+2}$
14. context POS bigram: $\mathbf{pos}_j^{j+1} \Big|_{j=i-2}^{i+1}$
15. word + context POS: $w_i \wedge pos_{i\pm 1}$
16. context word + POS: $w_{i\pm 1} \wedge pos_i$

Lexicon Features (unlexicalized)

WordNet only

17. OOV: λ_i is not in WordNet as a unigram lemma $\wedge pos_i$
18. compound: non-punctuation lemma λ_i and the {previous, next} lemma in the sentence (if it is non-punctuation; an intervening hyphen is allowed) form an entry in WordNet, possibly separated by a hyphen or space
19. compound-hyphen: $pos_i = \text{HYPH} \wedge$ previous and next tokens form an entry in WordNet, possibly separated by a hyphen or space
20. ambiguity class: if content word unigram λ_i is in WordNet, the set of POS categories it can belong to; else pos_i if not a content POS \wedge the POS of the longest MW match to which λ_i belongs (if any) \wedge the position in that match (B or I)

For each multiword lexicon

21. lexicon name \wedge status of token i in the shortest path segmentation (O, B, or I) \wedge subcategory of lexical entry whose match includes token i , if matched \wedge whether the match is gappy
22. the above \wedge POS tags of the first and last matched tokens in the expression

Over all multiword lexicons

23. at least k lexicons contain a match that includes this token (if $n \geq 1$ matches, n active features)
24. at least k lexicons contain a match that includes this token, starts with a given POS, and ends with a given POS

References

- Anne Abeillé, Lionel Clément, and François Toussnel. 2003. Building a treebank for French. In Anne Abeillé and Nancy Ide, editors, *Treebanks*, volume 20 of *Text, Speech and Language Technology*, pages 165–187. Kluwer Academic Publishers, Dordrecht, The Netherlands.
- Timothy Baldwin. 2005. Looking for prepositional verbs in corpus data. In *Proc. of the Second ACL-SIGSEM Workshop on the Linguistic Dimensions of Prepositions and their Use in Computational Linguistics Formalisms and Applications*, pages 115–126. Colchester, UK.
- Timothy Baldwin. 2008. A resource for evaluating the deep lexical acquisition of English verb-particle constructions. In *Proc. of MWE*, pages 1–2. Marrakech, Morocco.
- Timothy Baldwin and Su Nam Kim. 2010. Multiword expressions. In Nitin Indurkha and Fred J. Damerau, editors, *Handbook of Natural Language Processing, Second Edition*. CRC Press, Taylor and Francis Group, Boca Raton, Florida, USA.
- Eduard Bejček, Pavel Straňák, and Pavel Pecina. 2013. Syntactic identification of occurrences of multiword expressions in text using a lexicon with dependency structures. In *Proc. of the 9th Workshop on Multiword Expressions*, pages 106–115. Atlanta, Georgia, USA.
- Gábor Berend. 2011. Opinion expression mining by exploiting keyphrase extraction. In *Proc. of 5th International Joint Conference on Natural Language Processing*, pages 1162–1170. Chiang Mai, Thailand.
- Ann Bies, Justin Mott, Colin Warner, and Seth Kulick. 2012a. English Web Treebank. Technical Report LDC2012T13, Linguistic Data Consortium, Philadelphia, Pennsylvania, USA.
- Ann Bies, Justin Mott, Colin Warner, and Seth Kulick. 2012b. English Web Treebank. Technical Report LDC2012T13, Linguistic Data Consortium, Philadelphia, Pennsylvania, USA.
- Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O’Reilly Media, Inc., Sebastopol, California, USA.
- Phil Blunsom and Timothy Baldwin. 2006. Multilingual deep lexical acquisition for HPSGs via supertagging. In *Proc. of EMNLP*, pages 164–171. Sydney, Australia.
- Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.
- Marine Carpuat and Mona Diab. 2010. Task-based evaluation of multiword expressions: a pilot study in statistical machine translation. In *Proc. of NAACL-HLT*, pages 242–245. Los Angeles, California, USA.
- Kenneth Church, William Gale, Patrick Hanks, and Donald Hindle. 1991. Using statistics in lexical analysis. In Uri Zernik, editor, *Lexical acquisition: exploiting on-line resources to build a lexicon*, pages 115–164. Lawrence Erlbaum Associates, Hillsdale, New Jersey, USA.
- Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proc. of EMNLP*, pages 594–602. Sydney, Australia.
- Michael Collins. 2002. Discriminative training methods for Hidden Markov Models: theory and experiments with perceptron algorithms. In *Proc. of EMNLP*, pages 1–8. Philadelphia, Pennsylvania, USA.
- Mathieu Constant and Anthony Sigogne. 2011. MWU-aware part-of-speech tagging with a CRF model and lexical resources. In *Proc. of the Workshop on Multiword Expressions: from Parsing and Generation to the Real World*, pages 49–56. Portland, Oregon, USA.
- Mathieu Constant, Anthony Sigogne, and Patrick Watrin. 2012. Discriminative strategies to integrate multiword expression recognition and parsing. In *Proc. of ACL*, pages 204–212. Jeju Island, Korea.
- Paul Cook, Afsaneh Fazly, and Suzanne Stevenson. 2008. The VNC-Tokens dataset. In *Proc. of MWE*, pages 19–22. Marrakech, Morocco.
- Hal Daumé, III. 2006. *Practical structured learning techniques for natural language processing*. Ph.D. dissertation, University of Southern California, Los Angeles, California, USA. URL <http://hal3.name/docs/daume06thesis.pdf>.
- Mona Diab and Pravin Bhutada. 2009. Verb noun construction MWE token classification. In *Proc. of MWE*, pages 17–22. Suntec, Singapore.
- Nick C. Ellis, Rita Simpson-Vlach, and Carson Maynard. 2008. Formulaic language in native and second language speakers: psycholinguistics, corpus linguistics, and TESOL. *TESOL Quarterly*, 42(3):375–396.
- Christiane Fellbaum, editor. 1998. *WordNet: an electronic lexical database*. MIT Press, Cambridge, Massachusetts, USA.
- Charles J. Fillmore, Paul Kay, and Mary Catherine O’Connor. 1988. Regularity and idiomaticity in grammatical constructions: the case of ‘let alone’. *Language*, 64(3):501–538.
- Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.
- Mahmoud Ghoneim and Mona Diab. 2013. Multiword expressions in the context of statistical machine trans-

- lation. In *Proc. of IJCNLP*, pages 1181–1187. Nagoya, Japan.
- Kevin Gimpel and Noah A. Smith. 2011. Generative models of monolingual and bilingual gappy patterns. In *Proc. of WMT*, pages 512–522. Edinburgh, Scotland, UK.
- Adele E. Goldberg. 1995. *Constructions: a construction grammar approach to argument structure*. University of Chicago Press, Chicago, Illinois, USA.
- Adele E. Goldberg. 2006. *Constructions at work: the nature of generalization in language*. Oxford University Press, Oxford, UK.
- Edouard Grave, Guillaume Obozinski, and Francis Bach. 2013. Hidden Markov tree models for semantic class induction. In *Proc. of CoNLL*, pages 94–103. Sofia, Bulgaria.
- Spence Green, Marie-Catherine de Marneffe, John Bauer, and Christopher D. Manning. 2011. Multiword expression identification with tree substitution grammars: a parsing tour de force with French. In *Proc. of EMNLP*, pages 725–735. Edinburgh, Scotland, UK.
- Spence Green, Marie-Catherine de Marneffe, and Christopher D. Manning. 2012. Parsing models for identifying multiword expressions. *Computational Linguistics*, 39(1):195–227.
- Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semečký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Uřešová, and Zdeněk Žabokrtský. 2012. Prague Czech-English Dependency Treebank 2.0. Technical Report LDC2012T08, Linguistic Data Consortium, Philadelphia, Pennsylvania, USA. URL <http://www ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2003T10>.
- Silvana Hartmann, György Szarvas, and Iryna Gurevych. 2012. Mining multiword terms from Wikipedia. In Maria Teresa Pazienza and Armando Stellato, editors, *Semi-Automatic Ontology Development*. IGI Global, Hershey, Pennsylvania, USA.
- Chikara Hashimoto and Daisuke Kawahara. 2008. Construction of an idiom corpus and its application to idiom identification based on WSD incorporating idiom-specific features. In *Proc. of EMNLP*, pages 992–1001. Honolulu, Hawaii, USA.
- Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proc. of ACL-08: HLT*, pages 595–603. Columbus, Ohio.
- Koenraad Kuiper, Heather McCann, Heidi Quinn, Therese Aitchison, and Kees van der Veer. 2003. SAID. Technical Report LDC2003T10, Linguistic Data Consortium, Philadelphia, Pennsylvania, USA. URL <http://www ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2003T10>.
- John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*, pages 282–289.
- Percy Liang. 2005. *Semi-supervised learning for natural language*. Master’s thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA. URL <http://people.csail.mit.edu/pliang/papers/meng-thesis.pdf>.
- 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.
- I. Dan Melamed. 1997. Automatic discovery of non-compositional compounds in parallel data. In *Proc. of EMNLP*, pages 97–108. Providence, Rhode Island, USA.
- George A. Miller, Claudia Leacock, Randee Teng, and Ross T. Bunker. 1993. A semantic concordance. In *Proc. of HLT*, pages 303–308. Plainsboro, New Jersey, USA.
- Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *Proc. of HLT-NAACL*, pages 337–342. Boston, Massachusetts, USA.
- Behrang Mohit, Nathan Schneider, Rishav Bhowmick, Kemal Oflazer, and Noah A. Smith. 2012. Recall-oriented learning of named entities in Arabic Wikipedia. In *Proc. of EACL*, pages 162–173. Avignon, France.
- Begona Villada Moirón and Jörg Tiedemann. 2006. Identifying idiomatic expressions using automatic word-alignment. In *Proc. of the EACL 2006 Workshop on Multi-word Expressions in a Multilingual Context*, pages 33–40. Trento, Italy.
- Rosamund Moon. 1998. *Fixed expressions and idioms in English: a corpus-based approach*. Oxford Studies in Lexicography and Lexicology. Clarendon Press, Oxford, UK.
- David Newman, Nagendra Koilada, Jey Han Lau, and Timothy Baldwin. 2012. Bayesian text segmentation for index term identification and keyphrase extraction. In *Proc. of COLING 2012*, pages 2077–2092. Mumbai, India.
- Geoffrey Nunberg, Ivan A. Sag, and Thomas Wasow. 1994. Idioms. *Language*, 70(3):491–538.
- Olutobi Owoputi, Brendan O’Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proc. of NAACL-HLT*, pages 380–390. Atlanta, Georgia, USA.
- Gerhard Paaß and Frank Reichartz. 2009. Exploiting

- semantic constraints for estimating supersenses with CRFs. In *Proc. of the Ninth SIAM International Conference on Data Mining*, pages 485–496. Sparks, Nevada, USA.
- Pavel Pecina. 2010. Lexical association measures and collocation extraction. *Language Resources and Evaluation*, 44(1):137–158.
- Carlos Ramisch. 2012. *A generic and open framework for multiword expressions treatment: from acquisition to applications*. Ph.D. dissertation, University of Grenoble and Federal University of Rio Grande do Sul, Grenoble, France. URL http://www.inf.ufrgs.br/~ceramisch/download_files/thesis-getalp.pdf.
- Carlos Ramisch, Vitor De Araujo, and Aline Villavicencio. 2012. A broad evaluation of techniques for automatic acquisition of multiword expressions. In *Proc. of ACL 2012 Student Research Workshop*, pages 1–6. Jeju Island, Korea.
- Carlos Ramisch, Aline Villavicencio, and Christian Boitet. 2010. mwetoolkit: a framework for multiword expression identification. In *Proc. of LREC*, pages 662–669. Valletta, Malta.
- Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text chunking using transformation-based learning. In *Proc. of the Third ACL Workshop on Very Large Corpora*, pages 82–94. Cambridge, Massachusetts, USA.
- Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proc. of CoNLL*, pages 147–155. Boulder, Colorado, USA.
- Marta Recasens and Eduard Hovy. 2011. BLANC: Implementing the Rand index for coreference evaluation. *Natural Language Engineering*, 17(04):485–510.
- Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: an experimental study. In *Proc. of EMNLP*, pages 1524–1534. Edinburgh, Scotland, UK.
- Ivan Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: a pain in the neck for NLP. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 2276 of *Lecture Notes in Computer Science*, pages 189–206. Springer, Berlin, Germany.
- Nathan Schneider, Spencer Onuffer, Nora Kazour, Emily Danchik, Michael T. Mordowanec, Henrietta Conrad, and Noah A. Smith. 2014. Comprehensive annotation of multiword expressions in a social web corpus. In *Proc. of LREC*. Reykjavík, Iceland.
- Yutaro Shigeto, Ai Azuma, Sorami Hisamoto, Shuhei Kondo, Tomoya Kouse, Keisuke Sakaguchi, Akifumi Yoshimoto, Frances Yung, and Yuji Matsumoto. 2013. Construction of English MWE dictionary and its application to POS tagging. In *Proc. of the 9th Workshop on Multiword Expressions*, pages 139–144. Atlanta, Georgia, USA.
- James W. Thatcher. 1967. Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. *Journal of Computer and System Sciences*, 1(4):317–322.
- Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. of HLT-NAACL*, pages 173–180. Edmonton, Alberta, Canada.
- Yulia Tsvetkov and Shuly Wintner. 2010. Extraction of multi-word expressions from small parallel corpora. In *Coling 2010: Posters*, pages 1256–1264. Beijing, China.
- Yulia Tsvetkov and Shuly Wintner. 2011. Identification of multi-word expressions by combining multiple linguistic information sources. In *Proc. of EMNLP*, pages 836–845. Edinburgh, Scotland, UK.
- Yuancheng Tu and Dan Roth. 2011. Learning English light verb constructions: contextual or statistical. In *Proc. of the Workshop on Multiword Expressions: from Parsing and Generation to the Real World*, pages 31–39. Portland, Oregon, USA.
- Yuancheng Tu and Dan Roth. 2012. Sorting out the most confusing English phrasal verbs. In *Proc. of *SEM*, pages 65–69. Montréal, Quebec, Canada.
- Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proc. of ACL*, pages 384–394. Uppsala, Sweden.
- Martin Čmejrek, Jan Cuřín, Jan Hajič, and Jiří Havelka. 2005. Prague Czech-English Dependency Treebank: resource for structure-based MT. In *Proc. of EAMT*, pages 73–78. Budapest, Hungary.
- Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proc. of MUC-6*, pages 45–52. Columbia, Maryland, USA.
- Veronika Vincze. 2012. Light verb constructions in the SzegedParalellFX English-Hungarian parallel corpus. In *Proc. of LREC*. Istanbul, Turkey.
- Veronika Vincze, István Nagy T., and János Zsibrita. 2013. Learning to detect English and Hungarian light verb constructions. *ACM Transactions on Speech and Language Processing*, 10(2):6:1–6:25.

Segmentation for Efficient Supervised Language Annotation with an Explicit Cost-Utility Tradeoff

Matthias Sperber¹, Mirjam Simantzik², Graham Neubig³, Satoshi Nakamura³, Alex Waibel¹

¹Karlsruhe Institute of Technology, Institute for Anthropomatics, Germany

²Mobile Technologies GmbH, Germany

³Nara Institute of Science and Technology, AHC Laboratory, Japan

matthias.sperber@kit.edu, mirjam.simantzik@jibbiggo.com, neubig@is.naist.jp
s-nakamura@is.naist.jp, waibel@kit.edu

Abstract

In this paper, we study the problem of manually correcting automatic annotations of natural language in as efficient a manner as possible. We introduce a method for automatically segmenting a corpus into chunks such that many uncertain labels are grouped into the same chunk, while human supervision can be omitted altogether for other segments. A tradeoff must be found for segment sizes. Choosing short segments allows us to reduce the number of highly confident labels that are supervised by the annotator, which is useful because these labels are often already correct and supervising correct labels is a waste of effort. In contrast, long segments reduce the cognitive effort due to context switches. Our method helps find the segmentation that optimizes supervision efficiency by defining user models to predict the cost and utility of supervising each segment and solving a constrained optimization problem balancing these contradictory objectives. A user study demonstrates noticeable gains over pre-segmented, confidence-ordered baselines on two natural language processing tasks: speech transcription and word segmentation.

1 Introduction

Many natural language processing (NLP) tasks require human supervision to be useful in practice, be it to collect suitable training material or to meet some desired output quality. Given the high cost of human intervention, how to minimize the supervision effort is an important research problem. Previous works in areas such as active learning, post edit-

- | |
|--|
| (a) It was a bright cold <u>(they)</u> in <u>(apron)</u> , and <u>(a)</u> clocks were striking thirteen. |
| (b) It was a bright cold <u>(they)</u> in <u>(apron)</u> , and <u>(a)</u> clocks were striking thirteen. |
| (c) It was a bright cold <u>(they)</u> in <u>(apron)</u> , and <u>(a)</u> clocks were striking thirteen. |

Figure 1: Three automatic transcripts of the sentence “It was a bright cold day in April, and the clocks were striking thirteen”, with recognition errors in parentheses. The underlined parts are to be corrected by a human for (a) sentences, (b) words, or (c) the proposed segmentation.

ing, and interactive pattern recognition have investigated this question with notable success (Settles, 2008; Specia, 2011; González-Rubio et al., 2010).

The most common framework for efficient annotation in the NLP context consists of training an NLP system on a small amount of baseline data, and then running the system on unannotated data to estimate confidence scores of the system’s predictions (Settles, 2008). Sentences with the lowest confidence are then used as the data to be annotated (Figure 1 (a)). However, it has been noted that when the NLP system in question already has relatively high accuracy, annotating entire sentences can be wasteful, as most words will already be correct (Tomanek and Hahn, 2009; Neubig et al., 2011). In these cases, it is possible to achieve much higher benefit per annotated word by annotating sub-sentential units (Figure 1 (b)).

However, as Settles et al. (2008) point out, simply maximizing the benefit per annotated instance is not enough, as the real supervision effort varies

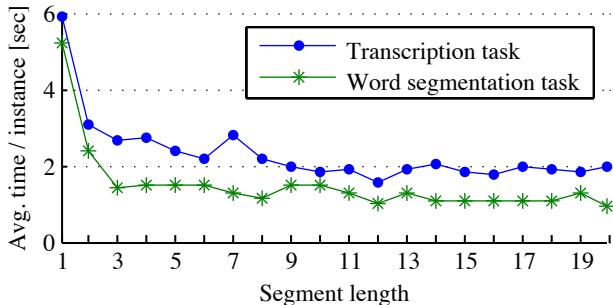


Figure 2: Average annotation time per instance, plotted over different segment lengths. For both tasks, the effort clearly increases for short segments.

greatly across instances. This is particularly important in the context of choosing segments to annotate, as human annotators heavily rely on semantics and context information to process language, and intuitively, a consecutive sequence of words can be supervised faster and more accurately than the same number of words spread out over several locations in a text. This intuition can also be seen in our empirical data in Figure 2, which shows that for the speech transcription and word segmentation tasks described later in Section 5, short segments had a longer annotation time per word. Based on this fact, we argue it would be desirable to present the annotator with a segmentation of the data into easily supervisable chunks that are both large enough to reduce the number of context switches, and small enough to prevent unnecessary annotation (Figure 1 (c)).

In this paper, we introduce a new strategy for natural language supervision tasks that attempts to optimize supervision efficiency by choosing an appropriate segmentation. It relies on a user model that, given a specific segment, predicts the cost and the utility of supervising that segment. Given this user model, the goal is to find a segmentation that minimizes the total predicted cost while maximizing the utility. We balance these two criteria by defining a constrained optimization problem in which one criterion is the optimization objective, while the other criterion is used as a constraint. Doing so allows specifying practical optimization goals such as “remove as many errors as possible given a limited time budget,” or “annotate data to obtain some required classifier accuracy in as little time as possible.”

Solving this optimization task is computationally

difficult, an NP-hard problem. Nevertheless, we demonstrate that by making realistic assumptions about the segment length, an optimal solution can be found using an integer linear programming formulation for mid-sized corpora, as are common for supervised annotation tasks. For larger corpora, we provide simple heuristics to obtain an approximate solution in a reasonable amount of time.

Experiments over two example scenarios demonstrate the usefulness of our method: Post editing for speech transcription, and active learning for Japanese word segmentation. Our model predicts noticeable efficiency gains, which are confirmed in experiments with human annotators.

2 Problem Definition

The goal of our method is to find a segmentation over a corpus of word tokens w_1^N that optimizes supervision efficiency according to some predictive user model. The user model is denoted as a set of functions $u_{l,k}(w_a^b)$ that evaluate any possible subsequence w_a^b of tokens in the corpus according to criteria $l \in L$, and supervision modes $k \in K$.

Let us illustrate this with an example. Sperber et al. (2013) defined a framework for speech transcription in which an initial, erroneous transcript is created using automatic speech recognition (ASR), and an annotator corrects the transcript either by correcting the words by keyboard, by respeaking the content, or by leaving the words as is. In this case, we could define $K = \{\text{TYPE}, \text{RESPEAK}, \text{SKIP}\}$, each constant representing one of these three supervision modes. Our method will automatically determine the appropriate supervision mode for each segment.

The user model in this example might evaluate every segment according to two criteria L , a cost criterion (in terms of supervision time) and a utility criterion (in terms of number of removed errors), when using each mode. Intuitively, respeaking should be assigned both lower cost (because speaking is faster than typing), but also lower utility than typing on a keyboard (because respeaking recognition errors can occur). The SKIP mode denotes the special, unsupervised mode that always returns 0 cost and 0 utility.

Other possible supervision modes include multiple input modalities (Suhm et al., 2001), several human annotators with different expertise and cost

(Donmez and Carbonell, 2008), and correction vs. translation from scratch in machine translation (Specia, 2011). Similarly, cost could instead be expressed in monetary terms, or the utility function could predict the improvement of a classifier when the resulting annotation is not intended for direct human consumption, but as training data for a classifier in an active learning framework.

3 Optimization Framework

Given this setting, we are interested in simultaneously finding optimal locations and supervision modes for all segments, according to the given criteria. Each resulting segment will be assigned exactly one of these supervision modes. We denote a segmentation of the N tokens of corpus w_1^N into $M \leq N$ segments by specifying segment boundary markers $s_1^{M+1} = (s_1=1, s_2, \dots, s_{M+1}=N+1)$. Setting a boundary marker $s_i=a$ means that we put a segment boundary before the a -th word token (or the end-of-corpus marker for $a=N+1$). Thus our corpus is segmented into token sequences $[(w_{s_j}, \dots, w_{s_{j+1}-1})]_{j=1}^M$. The supervision modes assigned to each segment are denoted by m_j . We favor those segmentations that minimize the cumulative value $\sum_{j=1}^M [u_{l,m_j}(w_{s_j}^{s_{j+1}})]$ for each criterion l . For any criterion where larger values are intuitively better, we flip the sign before defining $u_{l,m_j}(w_{s_j}^{s_{j+1}})$ to maintain consistency (e.g. negative number of errors removed).

3.1 Multiple Criteria Optimization

In the case of a single criterion ($|L|=1$), we obtain a simple, single-objective unconstrained linear optimization problem, efficiently solvable via dynamic programming (Terzi and Tsaparas, 2006). However, in practice one usually encounters several competing criteria, such as cost and utility, and here we will focus on this more realistic setting. We balance competing criteria by using one as an optimization objective, and the others as constraints.¹ Let crite-

¹This approach is known as the *bounded objective function method* in multi-objective optimization literature (Marler and Arora, 2004). The very popular *weighted sum method* merges criteria into a single efficiency measure, but is problematic in our case because the number of supervised tokens is unspecified. Unless the weights are carefully chosen, the algorithm might find, e.g., the completely unsupervised or completely su-

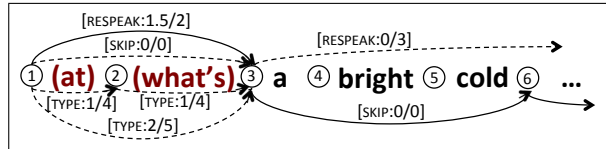


Figure 3: Excerpt of a segmentation graph for an example transcription task similar to Figure 1 (some edges are omitted for readability). Edges are labeled with their mode, predicted number of errors that can be removed, and necessary supervision time. A segmentation scheme might prefer solid edges over dashed ones in this example.

riterion l' be the optimization objective criterion, and let C_l denote the constraining constants for the criteria $l \in L_{-l'} = L \setminus \{l'\}$. We state the optimization problem:

$$\begin{aligned} \min_{M; s_1^{M+1}; m_1^M} & \sum_{j=1}^M [u_{l',m_j}(w_{s_j}^{s_{j+1}})] \\ \text{s.t.} & \sum_{j=1}^M [u_{l,m_j}(w_{s_j}^{s_{j+1}})] \leq C_l \quad (\forall l \in L_{-l'}) \end{aligned}$$

This constrained optimization problem is difficult to solve. In fact, the NP-hard multiple-choice knapsack problem (Pisinger, 1994) corresponds to a special case of our problem in which the number of segments is equal to the number of tokens, implying that our more general problem is NP-hard as well.

In order to overcome this problem, we reformulate search for the optimal segmentation as a resource-constrained shortest path problem in a directed, acyclic multigraph. While still not efficiently solvable in theory, this problem is well studied in domains such as vehicle routing and crew scheduling (Irnich and Desaulniers, 2005), and it is known that in many practical situations the problem can be solved reasonably efficiently using integer linear programming relaxations (Toth and Vigo, 2001).

In our formalism, the set of nodes V represents the spaces between neighboring tokens, at which the algorithm may insert segment boundaries. A node with index i represents a segment break before the i -th token, and thus the sequence of the indices in a path directly corresponds to s_1^{M+1} . Edges E denote the grouping of tokens between the respective supervised segmentation to be most "efficient."

nodes into one segment. Edges are always directed from left to right, and labeled with a supervision mode. In addition, each edge between nodes i and j is assigned $u_{l,k}(w_i^{j-1})$, the corresponding predicted value for each criterion $l \in L$ and supervision mode $k \in K$, indicating that the supervision mode of the j -th segment in a path directly corresponds to m_j .

Figure 3 shows an example of what the resulting graph may look like. Our original optimization problem is now equivalent to finding the shortest path between the first and last nodes according to criterion l' , while obeying the given resource constraints. According to a widely used formulation for the resource constrained shortest path problem, we can define E_{ij} as the set of competing edges between i and j , and express this optimization problem with the following integer linear program (ILP):

$$\min_{\mathbf{x}} \sum_{i,j \in V} \sum_{k \in E_{ij}} x_{ijk} u_{l',k}(s_i^{j-1}) \quad (1)$$

$$\text{s.t.} \sum_{i,j \in V} \sum_{k \in E_{ij}} x_{ijk} u_{l,k}(s_i^{j-1}) \leq C_l \quad (2)$$

$$(\forall l \in L_{-l'})$$

$$\sum_{\substack{i \in V \\ k \in E_{ij}}} x_{ijk} = \sum_{\substack{i \in V \\ k \in E_{ij}}} x_{jik} \quad (3)$$

$$(\forall j \in V \setminus \{1, n\})$$

$$\sum_{\substack{j \in V \\ k \in E_{1j}}} x_{1jk} = 1 \quad (4)$$

$$\sum_{\substack{i \in V \\ k \in E_{in}}} x_{ink} = 1 \quad (5)$$

$$x_{ijk} \in \{0, 1\} \quad (\forall x_{ijk} \in \mathbf{x}) \quad (6)$$

The variables $\mathbf{x} = \{x_{ijk} | i, j \in V, k \in E_{ij}\}$ denote the activation of the k 'th edge between nodes i and j . The shortest path according to the minimization objective (1), that still meets the resource constraints for the specified criteria (2), is to be computed. The degree constraints (3,4,5) specify that all but the first and last nodes must have as many incoming as outgoing edges, while the first node must have exactly one outgoing, and the last node exactly one incoming edge. Finally, the integrality condition (6) forces all edges to be either fully activated or fully deactivated. The outlined problem formulation can solved

directly by using off-the-shelf ILP solvers, here we employ GUROBI (Gurobi Optimization, 2012).

3.2 Heuristics for Approximation

In general, edges are inserted for every supervision mode between every combination of two nodes. The search space can be constrained by removing some of these edges to increase efficiency. In this study, we only consider edges spanning at most 20 tokens.

For cases in which larger corpora are to be annotated, or when the acceptable delay for delivering results is small, a suitable segmentation can be found approximately. The easiest way would be to partition the corpus, e.g. according to its individual documents, divide the budget constraints evenly across all partitions, and then segment each partition independently. More sophisticated methods might approximate the Pareto front for each partition, and distribute the budgets in an intelligent way.

4 User Modeling

While the proposed framework is able to optimize the segmentation with respect to each criterion, it also rests upon the assumption that we can provide user models $u_{l,k}(w_i^{j-1})$ that accurately evaluate every segment according to the specified criteria and supervision modes. In this section, we discuss our strategies for estimating three conceivable criteria: annotation cost, correction of errors, and improvement of a classifier.

4.1 Annotation Cost Modeling

Modeling cost requires solving a regression problem from features of a candidate segment to annotation cost, for example in terms of supervision time. Appropriate input features depend on the task, but should include notions of complexity (e.g. a confidence measure) and length of the segment, as both are expected to strongly influence supervision time.

We propose using Gaussian process (GP) regression for cost prediction, a start-of-the-art nonparametric Bayesian regression technique (Rasmussen and Williams, 2006)². As reported on a similar task by Cohn and Specia (2013), and confirmed by our preliminary experiments, GP regression significantly outperforms popular techniques such as sup-

²Code available at <http://www.gaussianprocess.org/gpml/>

port vector regression and least-squares linear regression. We also follow their settings for GP, employing GP regression with a squared exponential kernel with automatic relevance determination. Depending on the number of users and amount of training data available for each user, models may be trained separately for each user (as we do here), or in a combined fashion via multi-task learning as proposed by Cohn and Specia (2013).

It is also crucial for the predictions to be reliable throughout the whole relevant space of segments. If the cost of certain types of segments is systematically underpredicted, the segmentation algorithm might be misled to prefer these, possibly a large number of times.³ An effective trick to prevent such underpredictions is to predict the log time instead of the actual time. In this way, errors in the critical low end are penalized more strongly, and the time can never become negative.

4.2 Error Correction Modeling

As one utility measure, we can use the number of errors corrected, a useful measure for post editing tasks over automatically produced annotations. In order to measure how many errors can be removed by supervising a particular segment, we must estimate both how many errors are in the automatic annotation, and how reliably a human can remove these for a given supervision mode.

Most machine learning techniques can estimate confidence scores in the form of posterior probabilities. To estimate the number of errors, we can sum over one minus the posterior for all tokens, which estimates the Hamming distance from the reference annotation. This measure is appropriate for tasks in which the number of tokens is fixed in advance (e.g. a part-of-speech estimation task), and a reasonable approximation for tasks in which the number of tokens is not known in advance (e.g. speech transcription, cf. Section 5.1.1).

Predicting the particular tokens at which a human will make a mistake is known to be a difficult task (Olson and Olson, 1990), but a simplifying constant

³For instance, consider a model that predicts well for segments of medium size or longer, but underpredicts the supervision time of single-token segments. This may lead the segmentation algorithm to put every token into its own segment, which is clearly undesirable.

human error rate can still be useful. For example, in the task from Section 2, we may suspect a certain number of errors in a transcript segment, and predict, say, 95% of those errors to be removed via typing, but only 85% via respeaking.

4.3 Classifier Improvement Modeling

Another reasonable utility measure is accuracy of a classifier trained on the data we choose to annotate in an active learning framework. Confidence scores have been found useful for ranking particular tokens with regards to how much they will improve a classifier (Settles, 2008). Here, we may similarly score segment utility as the sum of its token confidences, although care must be taken to normalize and calibrate the token confidences to be linearly comparable before doing so. While the resulting utility score has no interpretation in absolute terms, it can still be used as an optimization objective (cf. Section 5.2.1).

5 Experiments

In this section, we present experimental results examining the effectiveness of the proposed method over two tasks: speech transcription and Japanese word segmentation.⁴

5.1 Speech Transcription Experiments

Accurate speech transcripts are a much-demanded NLP product, useful by themselves, as training material for ASR, or as input for follow-up tasks like speech translation. With recognition accuracies plateauing, manually correcting (post editing) automatic speech transcripts has become popular. Common approaches are to identify words (Sanchez-Cortina et al., 2012) or (sub-)sentences (Sperber et al., 2013) of low confidence, and have a human editor correct these.

5.1.1 Experimental Setup

We conduct a user study in which participants post-edited speech transcripts, given a fixed goal word error rate. The transcription setup was such that the transcriber could see the ASR transcript of parts before and after the segment that he was editing, providing context if needed. When imprecise time alignment resulted in segment breaks that were

⁴Software and experimental data can be downloaded from <http://www.msperber.com/research/tacl-segmentation/>

slightly “off,” as happened occasionally, that context helped guess what was said. The segment itself was transcribed from scratch, as opposed to editing the ASR transcript; besides being arguably more efficient when the ASR transcript contains many mistakes (Nanjo et al., 2006; Akita et al., 2009), preliminary experiments also showed that supervision time is far easier to predict this way. Figure 4 illustrates what the setup looked like.

We used a self-developed transcription tool to conduct experiments. It presents our computed segments one by one, allows convenient input and playback via keyboard shortcuts, and logs user interactions with their time stamps. A selection of TED talks⁵ (English talks on technology, entertainment, and design) served as experimental data. While some of these talks contain jargon such as medical terms, they are presented by skilled speakers, making them comparably easy to understand. Initial transcripts were created using the Janus recognition toolkit (Soltau et al., 2001) with a standard, TED-optimized setup. We used confusion networks for decoding and obtaining confidence scores.

For reasons of simplicity, and better comparability to our baseline, we restricted our experiment to two supervision modes: `TYPE` and `SKIP`. We conducted experiments with 3 participants, 1 with several years of experience in transcription, 2 with none. Each participant received an explanation on the transcription guidelines, and a short hands-on training to learn to use our tool. Next, they transcribed a balanced selection of 200 segments of varying length and quality in random order. This data was used to train the user models.

Finally, each participant transcribed another 2 TED talks, with word error rate (WER) 19.96% (predicted: 22.33%). We set a target (predicted) WER of 15% as our optimization constraint,⁶ and minimize the predicted supervision time as our objective function. Both TED talks were transcribed once using the baseline strategy, and once using the proposed strategy. The order of both strategies was reversed between talks, to minimize learning bias due to transcribing each talk twice.

The baseline strategy was adopted according to

⁵www.ted.com

⁶Depending on the level of accuracy required by our final application, this target may be set lower or higher.

Sperber et al. (2013): We segmented the talk into natural, subsentential units, using Matusov et al. (2006)’s segmenter, which we tuned to reproduce the TED subtitle segmentation, producing a mean segment length of 8.6 words. Segments were added in order of increasing average word confidence, until the user model predicted a $WER < 15\%$. The second segmentation strategy was the proposed method, similarly with a resource constraint of $WER < 15\%$.

Supervision time was predicted via GP regression (cf. Section 4.1), using segment length, audio duration, and mean confidence as input features. The output variable was assumed subject to additive Gaussian noise with zero mean, a variance of 5 seconds was chosen empirically to minimize the mean squared error. Utility prediction (cf. Section 4.2) was based on posterior scores obtained from the confusion networks. We found it important to calibrate them, as the posteriors were overconfident especially in the upper range. To do so, we automatically transcribed a development set of TED data, grouped the recognized words into buckets according to their posteriors, and determined the average number of errors per word in each bucket from an alignment with the reference transcript. The mapping from average posterior to average number of errors was estimated via GP regression. The result was summed over all tokens, and multiplied by a constant human confidence, separately determined for each participant.⁷

5.1.2 Simulation Results

To convey a better understanding of the potential gains afforded by our method, we first present a simulated experiment. We assume a transcriber who makes no mistakes, and needs exactly the amount of time predicted by a user model trained on the data of a randomly selected participant. We compare three scenarios: A baseline simulation, in which the baseline segments are transcribed in ascending order of confidence; a simulation using the proposed method, in which we change the WER constraint in small increments; finally, an oracle simulation, which uses

⁷More elaborate methods for WER estimation exist, such as by Ogawa et al. (2013), but if our method achieves improvements using simple Hamming distance, incorporating more sophisticated measures will likely achieve similar, or even better accuracy.

(3) SKIP: “nineteen forty six until today you see the green”
(4) TYPE: <annotator types: “is the traditional”>
(5) SKIP: “Interstate conflict”
(6) TYPE: <annotator types: “the ones we used to”>
(7) SKIP: ...

Figure 4: Result of our segmentation method (excerpt). TYPE segments are displayed empty and should be transcribed from scratch. For SKIP segments, the ASR transcript is displayed to provide context. When annotating a segment, the corresponding audio is played back.

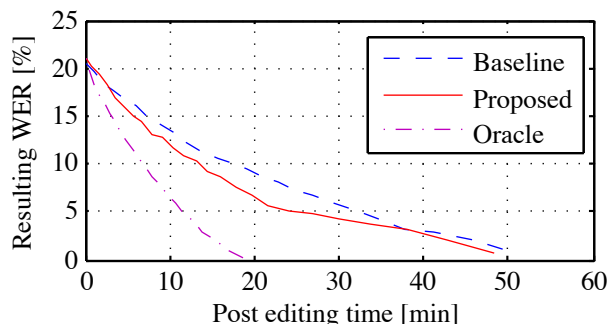


Figure 5: Simulation of post editing on example TED talk. The proposed method reduces the WER considerably faster than the baseline at first, later both converge. The much superior oracle simulation indicates room for further improvement.

the proposed method, but uses a utility model that knows the actual number of errors in each segment. For each supervised segment, we simply replace the ASR output with the reference, and measure the resulting WER.

Figure 5 shows the simulation on an example TED talk, based on an initial transcript with 21.9% WER. The proposed method is able to reduce the WER faster than the baseline, up to a certain point where they converge. The oracle simulation is even faster, indicating room for improvement through better confidence scores.

5.1.3 User Study Results

Table 1 shows the results of the user study. First, we note that the WER estimation by our utility model was off by about 2.5%: While the predicted improvement in WER was from 22.33% to 15.0%, the actual improvement was from 19.96% to about 12.5%. The actual resulting WER was consistent

Participant	Baseline		Proposed	
	WER	Time	WER	Time
P_1	12.26	44:05	12.18	33:01
P_2	12.75	36:19	12.77	29:54
P_3	12.70	52:42	12.50	37:57
AVG	12.57	44:22	12.48	33:37

Table 1: Transcription task results. For each user, the resulting WER [%] after supervision is shown, along with the time [min] they needed. The unsupervised WER was 19.96%.

across all users, and we observe strong, consistent reductions in supervision time for all participants. Prediction of the necessary supervision time was accurate: Averaged over participants, 45:41 minutes were predicted for the baseline, 44:22 minutes measured. For the proposed method, 32:11 minutes were predicted, 33:37 minutes measured. On average, participants removed 6.68 errors per minute using the baseline, and 8.93 errors per minute using the proposed method, a speed-up of 25.2%.

Note that predicted and measured values are not strictly comparable: In the experiments, to provide a fair comparison participants transcribed the same talks twice (once using baseline, once the proposed method, in alternating order), resulting in a noticeable learning effect. The user model, on the other hand, is trained to predict the case in which a transcriber conducts only one transcription pass.

As an interesting finding, without being informed about the order of baseline and proposed method, participants reported that transcribing according to the proposed segmentation seemed harder, as they found the baseline segmentation more linguistically reasonable. However, this perceived increase in difficulty did not show in efficiency numbers.

5.2 Japanese Word Segmentation Experiments

Word segmentation is the first step in NLP for languages that are commonly written without word boundaries, such as Japanese and Chinese. We apply our method to a task in which we domain-adapt a word segmentation classifier via active learning. In this experiment, participants annotated whether or not a word boundary occurred at certain positions in a Japanese sentence. The tokens to be grouped into segments are positions between adjacent characters.

5.2.1 Experimental Setup

Neubig et al. (2011) have proposed a pointwise method for Japanese word segmentation that can be trained using partially annotated sentences, which makes it attractive in combination with active learning, as well as our segmentation method. The authors released their method as a software package “KyTea” that we employed in this user study. We used KyTea’s active learning domain adaptation toolkit⁸ as a baseline.

For data, we used the Balanced Corpus of Contemporary Written Japanese (BCCWJ), created by Maekawa (2008), with the *internet Q&A subcorpus* as in-domain data, and the *whitepaper subcorpus* as background data, a domain adaptation scenario. Sentences were drawn from the in-domain corpus, and the manually annotated data was then used to train KyTea, along with the pre-annotated background data. The goal (objective function) was to improve KyTea’s classification accuracy on an in-domain test set, given a constrained time budget of 30 minutes. There were again 2 supervision modes: ANNOTATE and SKIP. Note that this is essentially a batch active learning setup with only one iteration.

We conducted experiments with one expert with several years of experience with Japanese word segmentation annotation, and three non-expert native speakers with no prior experience. Japanese word segmentation is not a trivial task, so we provided non-experts with training, including explanation of the segmentation standard, a supervised test with immediate feedback and explanations, and hands-on training to get used to the annotation software.

Supervision time was predicted via GP regression (cf. Section 4.1), using the segment length and mean confidence as input features. As before, the output variable was assumed subject to additive Gaussian noise with zero mean and 5 seconds variance. To obtain training data for these models, each participant annotated about 500 example instances, drawn from the adaptation corpus, grouped into segments and balanced regarding segment length and difficulty.

For utility modeling (cf. Section 4.3), we first normalized KyTea’s confidence scores, which are given in terms of SVM margin, using a sigmoid function (Platt, 1999). The normalization parameter was se-

lected so that the mean confidence on a development set corresponded to the actual classifier accuracy. We derive our measure of classifier improvement for correcting a segment by summing over one minus the calibrated confidence for each of its tokens. To analyze how well this measure describes the actual training utility, we trained KyTea using the background data plus disjoint groups of 100 in-domain instances with similar probabilities and measured the achieved reduction of prediction errors. The correlation between each group’s mean utility and the achieved error reduction was 0.87. Note that we ignore the decaying returns usually observed as more data is added to the training set. Also, we did not attempt to model user errors. Employing a constant base error rate, as in the transcription scenario, would change segment utilities only by a constant factor, without changing the resulting segmentation.

After creating the user models, we conducted the main experiment, in which each participant annotated data that was selected from a pool of 1000 in-domain sentences using two strategies. The first, baseline strategy was as proposed by Neubig et al. (2011). Queries are those instances with the lowest confidence scores. Each query is then extended to the left and right, until a word boundary is predicted. This strategy follows similar reasoning as was the premise to this paper: To decide whether or not a position in a text corresponds to a word boundary, the annotator has to acquire surrounding context information. This context acquisition is relatively time consuming, so he might as well label the surrounding instances with little additional effort. The second strategy was our proposed, more principled approach. Queries of both methods were shuffled to minimize bias due to learning effects. Finally, we trained KyTea using the results of both methods, and compared the achieved classifier improvement and supervision times.

5.2.2 User Study Results

Table 2 summarizes the results of our experiment. It shows that the annotations by each participant resulted in a better classifier for the proposed method than the baseline, but also took up considerably more time, a less clear improvement than for the transcription task. In fact, the total error for time predictions was as high as 12.5% on average,

⁸<http://www.phontron.com/kytea/active.html>

Participant	Baseline		Proposed	
	Time	Acc.	Time	Acc.
Expert	25:50	96.17	32:45	96.55
NonExp ₁	22:05	95.79	26:44	95.98
NonExp ₂	23:37	96.15	31:28	96.21
NonExp ₃	25:23	96.38	33:36	96.45

Table 2: Word segmentation task results, for our expert and 3 non-expert participants. For each participant, the resulting classifier accuracy [%] after supervision is shown, along with the time [min] they needed. The unsupervised accuracy was 95.14%.

where the baseline method tended take less time than predicted, the proposed method more time. This is in contrast to a much lower total error (within 1%) when cross-validating our user model training data. This is likely due to the fact that the data for training the user model was selected in a balanced manner, as opposed to selecting difficult examples, as our method is prone to do. Thus, we may expect much better predictions when selecting user model training data that is more similar to the test case.

Plotting classifier accuracy over annotation time draws a clearer picture. Let us first analyze the results for the expert annotator. Figure 6 (E.1) shows that the proposed method resulted in consistently better results, indicating that time predictions were still effective. Note that this comparison may put the proposed method at a slight disadvantage by comparing intermediate results despite optimizing globally.

For the non-experts, the improvement over the baseline is less consistent, as can be seen in Figure 6 (N.1) for one representative. According to our analysis, this can be explained by two factors: (1) The non-experts’ annotation error (6.5% on average) was much higher than the expert’s (2.7%), resulting in a somewhat irregular classifier learning curve. (2) The variance in annotation time per segment was consistently higher for the non-experts than the expert, indicated by an average per-segment prediction error of 71% vs. 58% relative to the mean actual value, respectively. Informally speaking, non-experts made more mistakes, and were more strongly influenced by the difficulty of a particular segment (which was higher on average with the proposed method, as indicated by a

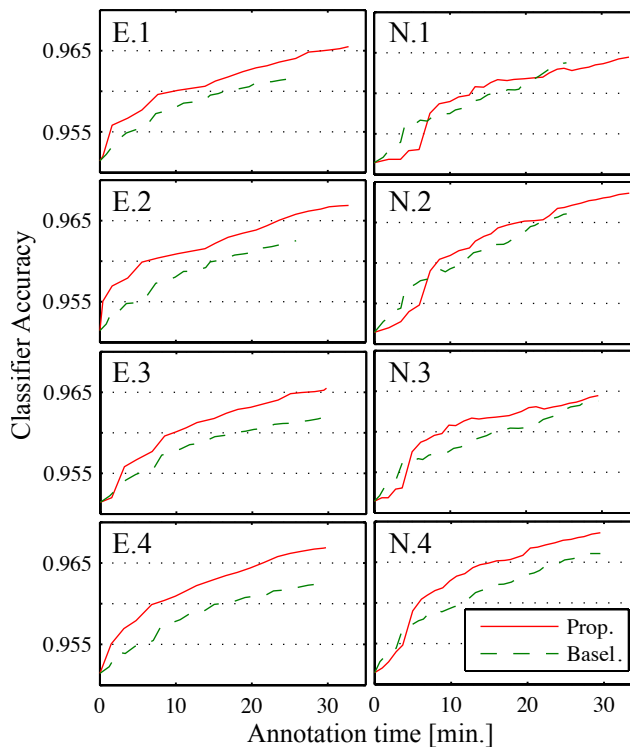


Figure 6: Classifier improvement over time, depicted for the expert (E) and a non-expert (N). The graphs show numbers based on (1) actual annotations and user models as in Sections 4.1 and 4.3, (2) error-free annotations, (3) measured times replaced by predicted times, and (4) both reference annotations and replaced time predictions.

lower average confidence).⁹

In Figures 6 (2-4) we present a simulation experiment in which we first pretend as if annotators made no mistakes, then as if they needed exactly as much time as predicted for each segment, and then both. This cheating experiment works in favor of the proposed method, especially for the non-expert. We may conclude that our segmentation approach is effective for the word segmentation task, but requires more accurate time predictions. Better user models will certainly help, although for the presented scenario our method may be most useful for an expert annotator.

⁹Note that the non-expert in the figure annotated much faster than the expert, which explains the comparable classification result despite making more annotation errors. This is in contrast to the other non-experts, who were slower.

5.3 Computational Efficiency

Since our segmentation algorithm does not guarantee polynomial runtime, computational efficiency was a concern, but did not turn out problematic. On a consumer laptop, the solver produced segmentations within a few seconds for a single document containing several thousand tokens, and within hours for corpora consisting of several dozen documents. Runtime increased roughly quadratically with respect to the number of segmented tokens. We feel that this is acceptable, considering that the time needed for human supervision will likely dominate the computation time, and reasonable approximations can be made as noted in Section 3.2.

6 Relation to Prior Work

Efficient supervision strategies have been studied across a variety of NLP-related research areas, and received increasing attention in recent years. Examples include post editing for speech recognition (Sanchez-Cortina et al., 2012), interactive machine translation (González-Rubio et al., 2010), active learning for machine translation (Haffari et al., 2009; González-Rubio et al., 2011) and many other NLP tasks (Olsson, 2009), to name but a few studies.

It has also been recognized by the active learning community that correcting the most useful parts first is often not optimal in terms of efficiency, since these parts tend to be the most difficult to manually annotate (Settles et al., 2008). The authors advocate the use of a user model to predict the supervision effort, and select the instances with best “bang-for-the-buck.” This prediction of supervision effort was successful, and was further refined in other NLP-related studies (Tomanek et al., 2010; Specia, 2011; Cohn and Specia, 2013). Our approach to user modeling using GP regression is inspired by the latter.

Most studies on user models consider only supervision effort, while neglecting the accuracy of human annotations. The view on humans as a perfect oracle has been criticized (Donmez and Carbonell, 2008), since human errors are common and can negatively affect supervision utility. Research on human-computer-interaction has identified the modeling of human errors as very difficult (Olson and Olson, 1990), depending on factors such as user experience, cognitive load, user interface design, and

fatigue. Nevertheless, even the simple error model used in our post editing task was effective.

The active learning community has addressed the problem of balancing utility and cost in some more detail. The previously reported “bang-for-the-buck” approach is a very simple, greedy approach to combine both into one measure. A more theoretically founded scalar optimization objective is the net benefit (utility minus costs) as proposed by Vijayanarasimhan and Grauman (2009), but unfortunately is restricted to applications where both can be expressed in terms of the same monetary unit. Vijayanarasimhan et al. (2010) and Donmez and Carbonell (2008) use a more practical approach that specifies a constrained optimization problem by allowing only a limited time budget for supervision. Our approach is a generalization thereof and allows either specifying an upper bound on the predicted cost, or a lower bound on the predicted utility.

The main novelty of our presented approach is the explicit modeling and selection of segments of various sizes, such that annotation efficiency is optimized according to the specified constraints. While some works (Sassano and Kurohashi, 2010; Neubig et al., 2011) have proposed using subsentential segments, we are not aware of any previous work that explicitly optimizes that segmentation.

7 Conclusion

We presented a method that can effectively choose a segmentation of a language corpus that optimizes supervision efficiency, considering not only the actual usefulness of each segment, but also the annotation cost. We reported noticeable improvements over strong baselines in two user studies. Future user experiments with more participants would be desirable to verify our observations, and allow further analysis of different factors such as annotator expertise. Also, future research may improve the user modeling, which will be beneficial for our method.

Acknowledgments

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n 287658 Bridges Across the Language Divide (EU-BRIDGE).

References

- Yuya Akita, Masato Mimura, and Tatsuya Kawahara. 2009. Automatic Transcription System for Meetings of the Japanese National Congress. In *Interspeech*, pages 84–87, Brighton, UK.
- Trevor Cohn and Lucia Specia. 2013. Modelling Annotator Bias with Multi-task Gaussian Processes: An Application to Machine Translation Quality Estimation. In *Association for Computational Linguistics Conference (ACL)*, Sofia, Bulgaria.
- Pinar Donmez and Jaime Carbonell. 2008. Proactive Learning : Cost-Sensitive Active Learning with Multiple Imperfect Oracles. In *Conference on Information and Knowledge Management (CIKM)*, pages 619–628, Napa Valley, CA, USA.
- Jesús González-Rubio, Daniel Ortiz-Martínez, and Francisco Casacuberta. 2010. Balancing User Effort and Translation Error in Interactive Machine Translation Via Confidence Measures. In *Association for Computational Linguistics Conference (ACL), Short Papers Track*, pages 173–177, Uppsala, Sweden.
- Jesús González-Rubio, Daniel Ortiz-Martínez, and Francisco Casacuberta. 2011. An active learning scenario for interactive machine translation. In *International Conference on Multimodal Interfaces (ICMI)*, pages 197–200, Alicante, Spain.
- Gurobi Optimization. 2012. Gurobi Optimizer Reference Manual.
- Gholamreza Haffari, Maxim Roy, and Anoop Sarkar. 2009. Active Learning for Statistical Phrase-based Machine Translation. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies Conference (NAACL-HLT)*, pages 415–423, Boulder, CO, USA.
- Stefan Irnich and Guy Desaulniers. 2005. Shortest Path Problems with Resource Constraints. In *Column Generation*, pages 33–65. Springer US.
- Kikuo Maekawa. 2008. Balanced Corpus of Contemporary Written Japanese. In *International Joint Conference on Natural Language Processing (IJCNLP)*, pages 101–102, Hyderabad, India.
- R. Timothy Marler and Jasbir S. Arora. 2004. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26(6):369–395, April.
- Evgeny Matusov, Arne Mauser, and Hermann Ney. 2006. Automatic Sentence Segmentation and Punctuation Prediction for Spoken Language Translation. In *International Workshop on Spoken Language Translation (IWSLT)*, pages 158–165, Kyoto, Japan.
- Hiroaki Nanjo, Yuya Akita, and Tatsuya Kawahara. 2006. Computer Assisted Speech Transcription System for Efficient Speech Archive. In *Western Pacific Acoustics Conference (WESPAC)*, Seoul, Korea.
- Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise Prediction for Robust , Adaptable Japanese Morphological Analysis. In *Association for Computational Linguistics: Human Language Technologies Conference (ACL-HLT)*, pages 529–533, Portland, OR, USA.
- Atsunori Ogawa, Takaaki Hori, and Atsushi Nakamura. 2013. Discriminative Recognition Rate Estimation For N-Best List and Its Application To N-Best Rescoring. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 6832–6836, Vancouver, Canada.
- Judith Reitman Olson and Gary Olson. 1990. The Growth of Cognitive Modeling in Human-Computer Interaction Since GOMS. *Human-Computer Interaction*, 5(2):221–265, June.
- Fredrik Olsson. 2009. A literature survey of active machine learning in the context of natural language processing. Technical report, SICS Sweden.
- David Pisinger. 1994. A Minimal Algorithm for the Multiple-Choice Knapsack Problem. *European Journal of Operational Research*, 83(2):394–410.
- John C. Platt. 1999. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press.
- Carl E. Rasmussen and Christopher K.I. Williams. 2006. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, USA.
- Isaias Sanchez-Cortina, Nicolas Serrano, Alberto Sanchis, and Alfons Juan. 2012. A prototype for Interactive Speech Transcription Balancing Error and Supervision Effort. In *International Conference on Intelligent User Interfaces (IUI)*, pages 325–326, Lisbon, Portugal.
- Manabu Sassano and Sadao Kurohashi. 2010. Using Smaller Constituents Rather Than Sentences in Active Learning for Japanese Dependency Parsing. In *Association for Computational Linguistics Conference (ACL)*, pages 356–365, Uppsala, Sweden.
- Burr Settles, Mark Craven, and Lewis Friedland. 2008. Active Learning with Real Annotation Costs. In *Neural Information Processing Systems Conference (NIPS) - Workshop on Cost-Sensitive Learning*, Lake Tahoe, NV, United States.
- Burr Settles. 2008. An Analysis of Active Learning Strategies for Sequence Labeling Tasks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1070–1079, Honolulu, USA.
- Hagen Soltau, Florian Metze, Christian Fügen, and Alex Waibel. 2001. A One-Pass Decoder Based on Polymorphic Linguistic Context Assignment. In *Automatic Speech Recognition and Understanding Work-*

- shop (ASRU)*, pages 214–217, Madonna di Campiglio, Italy.
- Lucia Specia. 2011. Exploiting Objective Annotations for Measuring Translation Post-editing Effort. In *Conference of the European Association for Machine Translation (EAMT)*, pages 73–80, Nice, France.
- Matthias Sperber, Graham Neubig, Christian Fügen, Satoshi Nakamura, and Alex Waibel. 2013. Efficient Speech Transcription Through Respeaking. In *Interspeech*, pages 1087–1091, Lyon, France.
- Bernhard Suhm, Brad Myers, and Alex Waibel. 2001. Multimodal error correction for speech user interfaces. *Transactions on Computer-Human Interaction*, 8(1):60–98.
- Evimaria Terzi and Panayiotis Tsaparas. 2006. Efficient algorithms for sequence segmentation. In *SIAM Conference on Data Mining (SDM)*, Bethesda, MD, USA.
- Katrin Tomanek and Udo Hahn. 2009. Semi-Supervised Active Learning for Sequence Labeling. In *International Joint Conference on Natural Language Processing (IJCNLP)*, pages 1039–1047, Singapore.
- Katrin Tomanek, Udo Hahn, and Steffen Lohmann. 2010. A Cognitive Cost Model of Annotations Based on Eye-Tracking Data. In *Association for Computational Linguistics Conference (ACL)*, pages 1158–1167, Uppsala, Sweden.
- Paolo Toth and Daniele Vigo. 2001. *The Vehicle Routing Problem*. Society for Industrial & Applied Mathematics (SIAM), Philadelphia.
- Sudheendra Vijayanarasimhan and Kristen Grauman. 2009. Whats It Going to Cost You?: Predicting Effort vs. Informativeness for Multi-Label Image Annotations. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2262–2269, Miami Beach, FL, USA.
- Sudheendra Vijayanarasimhan, Prateek Jain, and Kristen Grauman. 2010. Far-sighted active learning on a budget for image and video recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3035–3042, San Francisco, CA, USA, June.

The Language Demographics of Amazon Mechanical Turk

Ellie Paylick¹ Matt Post² Ann Irvine² Dmitry Kachaev² Chris Callison-Burch^{1,2}

¹Computer and Information Science Department, University of Pennsylvania

²Human Language Technology Center of Excellence, Johns Hopkins University

Abstract

We present a large scale study of the languages spoken by bilingual workers on Mechanical Turk (MTurk). We establish a methodology for determining the language skills of anonymous crowd workers that is more robust than simple surveying. We validate workers' self-reported language skill claims by measuring their ability to correctly translate words, and by geolocating workers to see if they reside in countries where the languages are likely to be spoken. Rather than posting a one-off survey, we posted paid tasks consisting of 1,000 assignments to translate a total of 10,000 words in each of 100 languages. Our study ran for several months, and was highly visible on the MTurk crowdsourcing platform, increasing the chances that bilingual workers would complete it. Our study was useful both to create bilingual dictionaries and to act as census of the bilingual speakers on MTurk. We use this data to recommend languages with the largest speaker populations as good candidates for other researchers who want to develop crowdsourced, multilingual technologies. To further demonstrate the value of creating data via crowdsourcing, we hire workers to create bilingual parallel corpora in six Indian languages, and use them to train statistical machine translation systems.

1 Overview

Crowdsourcing is a promising new mechanism for collecting data for natural language processing research. Access to a fast, cheap, and flexible workforce allows us to collect new types of data, potentially enabling new language technologies. Because crowdsourcing platforms like Amazon Mechanical

Turk (MTurk) give researchers access to a world-wide workforce, one obvious application of crowdsourcing is the creation of multilingual technologies. With an increasing number of active crowd workers located outside of the United States, there is even the potential to reach fluent speakers of lower resource languages. In this paper, we investigate the feasibility of hiring language informants on MTurk by conducting the first large-scale demographic study of the languages spoken by workers on the platform.

There are several complicating factors when trying to take a census of workers on MTurk. The workers' identities are anonymized, and Amazon provides no information about their countries of origin or their language abilities. Posting a simple survey to have workers report this information may be inadequate, since (a) many workers may never see the survey, (b) many opt not to do one-off surveys since potential payment is low, and (c) validating the answers of respondents is not straightforward.

Our study establishes a methodology for determining the language demographics of anonymous crowd workers that is more robust than simple surveying. We ask workers what languages they speak and what country they live in, and validate their claims by measuring their ability to correctly translate words and by recording their geolocation. To increase the visibility and the desirability of our tasks, we post 1,000 assignments in each of 100 languages. These tasks each consist of translating 10 foreign words into English. Two of the 10 words have known translations, allowing us to validate that the workers' translations are accurate. We construct bilingual dictionaries with up to 10,000 entries, with the majority of entries being new.

Surveying thousands of workers allows us to analyze current speaker populations for 100 languages.

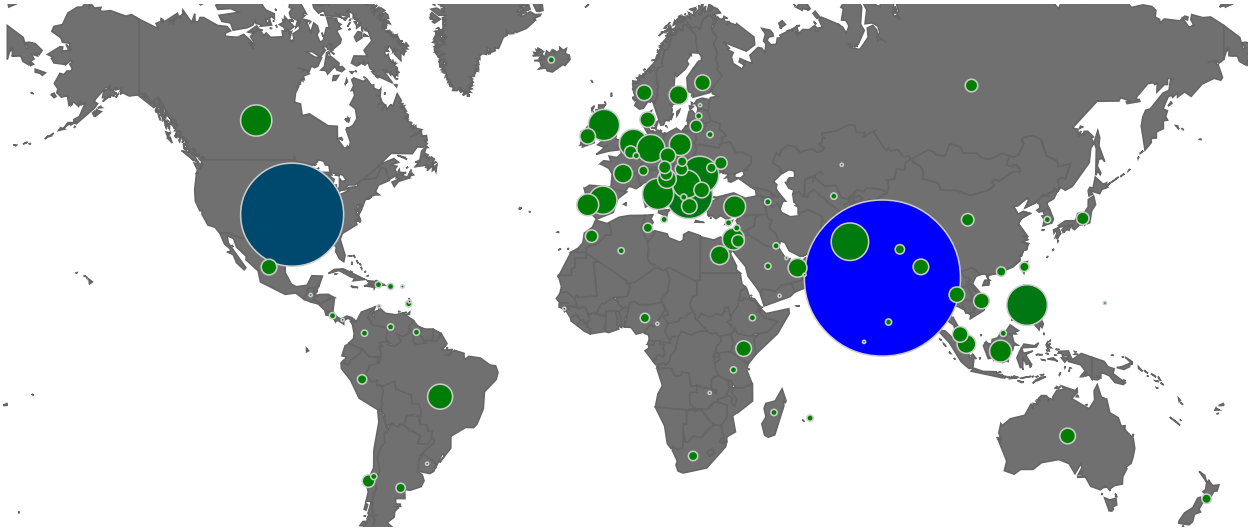


Figure 1: The number of workers per country. This map was generated based on geolocating the IP address of 4,983 workers in our study. Omitted are 60 workers who were located in more than one country during the study, and 238 workers who could not be geolocated. The size of the circles represents the number of workers from each country. The two largest are India (1,998 workers) and the United States (866). To calibrate the sizes: the Philippines has 142 workers, Egypt has 25, Russia has 10, and Sri Lanka has 4.

The data also allows us to answer questions like: How quickly is work completed in a given language? Are crowdsourced translations reliably good? How often do workers misrepresent their language abilities to obtain financial rewards?

2 Background and Related Work

Amazon’s Mechanical Turk (MTurk) is an on-line marketplace for work that gives employers and researchers access to a large, low-cost workforce. MTurk allows employers to provide micro-payments in return for workers completing micro-tasks. The basic units of work on MTurk are called ‘Human Intelligence Tasks’ (HITs). MTurk was designed to accommodate tasks that are difficult for computers, but simple for people. This facilitates research into human computation, where people can be treated as a function call (von Ahn, 2005; Little et al., 2009; Quinn and Bederson, 2011). It has application to research areas like human-computer interaction (Bigham et al., 2010; Bernstein et al., 2010), computer vision (Sorokin and Forsyth, 2008; Deng et al., 2010; Rashtchian et al., 2010), speech processing (Marge et al., 2010; Lane et al., 2010; Parent and Eskenazi, 2011; Eskenazi et al., 2013), and natural language processing (Snow et al., 2008; Callison-

Burch and Dredze, 2010; Laws et al., 2011).

On MTurk, researchers who need work completed are called ‘Requesters’, and workers are often referred to as ‘Turkers’. MTurk is a true market, meaning that Turkers are free to choose to complete the HITs which interest them, and Requesters can price their tasks competitively to try to attract workers and have their tasks done quickly (Faridani et al., 2011; Singer and Mittal, 2011). Turkers remain anonymous to Requesters, and all payment occurs through Amazon. Requesters are able to accept submitted work or reject work that does not meet their standards. Turkers are only paid if a Requester accepts their work.

Several reports examine Mechanical Turk as an economic market (Ipeirotis, 2010a; Lehdonvirta and Ernkvist, 2011). When Amazon introduced MTurk, it first offered payment only in Amazon credits, and later offered direct payment in US dollars. More recently, it has expanded to include one foreign currency, the Indian rupee. Despite its payments being limited to two currencies or Amazon credits, MTurk claims over half a million workers from 190 countries (Amazon, 2013). This suggests that its worker population should represent a diverse set of languages.

A demographic study by Ipeirotis (2010b) focused on age, gender, marital status, income levels, motivation for working on MTurk, and whether workers used it as a primary or supplemental form of income. The study contrasted Indian and US workers. Ross et al. (2010) completed a longitudinal follow-on study. A number of other studies have informally investigated Turkers’ language abilities. Munro and Tily (2011) compiled survey responses of 2,000 Turkers, revealing that four of the six most represented languages come from India (the top six being Hindi, Malayalam, Tamil, Spanish, French, and Telugu). Irvine and Klementiev (2010) had Turkers evaluate the accuracy of translations that had been automatically inducted from monolingual texts. They examined translations of 100 words in 42 low-resource languages, and reported geolocated countries for their workers (India, the US, Romania, Pakistan, Macedonia, Latvia, Bangladesh and the Philippines). Irvine and Klementiev discussed the difficulty of quality control and assessing the plausibility of workers’ language skills for rare languages, which we address in this paper.

Several researchers have investigated using MTurk to build bilingual parallel corpora for machine translation, a task which stands to benefit low cost, high volume translation on demand (Germann, 2001). Ambati et al. (2010) conducted a pilot study by posting 25 sentences to MTurk for Spanish, Chinese, Hindi, Telugu, Urdu, and Haitian Creole. In a study of 2000 Urdu sentences, Zaidan and Callison-Burch (2011) presented methods for achieving professional-level translation quality from Turkers by soliciting multiple English translations of each foreign sentence. Zbib et al. (2012) used crowdsourcing to construct a 1.5 million word parallel corpus of dialect Arabic and English, training a statistical machine translation system that produced higher quality translations of dialect Arabic than a system trained on 100 times more Modern Standard Arabic-English parallel data. Zbib et al. (2013) conducted a systematic study that showed that training an MT system on crowdsourced translations resulted in the same performance as training on professional translations, at $\frac{1}{5}$ the cost. Hu et al. (2010; Hu et al. (2011) performed crowdsourced translation by having monolingual speakers collaborate and iteratively improve MT output.

English	689	Tamil	253	Malayalam	219
Hindi	149	Spanish	131	Telugu	87
Chinese	86	Romanian	85	Portuguese	82
Arabic	74	Kannada	72	German	66
French	63	Polish	61	Urdu	56
Tagalog	54	Marathi	48	Russian	44
Italian	43	Bengali	41	Gujarati	39
Hebrew	38	Dutch	37	Turkish	35
Vietnamese	34	Macedonian	31	Cebuano	29
Swedish	26	Bulgarian	25	Swahili	23
Hungarian	23	Catalan	22	Thai	22
Lithuanian	21	Punjabi	21	Others	≤ 20

Table 1: Self-reported native language of 3,216 bilingual Turkers. Not shown are 49 languages with ≤ 20 speakers. We omit 1,801 Turkers who did not report their native language, 243 who reported 2 native languages, and 83 with ≥ 3 native languages.

Several researchers have examined cost optimization using active learning techniques to select the most useful sentences or fragments to translate (Ambati and Vogel, 2010; Bloodgood and Callison-Burch, 2010; Ambati, 2012).

To contrast our research with previous work, the main contributions of this paper are: (1) a robust methodology for assessing the bilingual skills of anonymous workers, (2) the largest-scale census to date of language skills of workers on MTurk, and (3) a detailed analysis of the data gathered in our study.

3 Experimental Design

The central task in this study was to investigate Mechanical Turk’s bilingual population. We accomplished this through self-reported surveys combined with a HIT to translate individual words for 100 languages. We evaluate the accuracy of the workers’ translations against known translations. In cases where these were not exact matches, we used a second pass monolingual HIT, which asked English speakers to evaluate if a worker-provided translation was a synonym of the known translation.

Demographic questionnaire At the start of each HIT, Turkers were asked to complete a brief survey about their language abilities. The survey asked the following questions:

- Is [language] your native language?
- How many years have you spoken [language]?

- Is English your native language?
- How many years have you spoken English?
- What country do you live in?

We automatically collected each worker’s current location by geolocating their IP address. A total of 5,281 unique workers completed our HITs. Of these, 3,625 provided answers to our survey questions, and we were able to geolocate 5,043. Figure 1 plots the location of workers across 106 countries. Table 1 gives the most common self-reported native languages.

Selection of languages We drew our data from the different language versions of Wikipedia. We selected the 100 languages with the largest number of articles ¹ (Table 2). For each language, we chose the 1,000 most viewed articles over a 1 year period,² and extracted the 10,000 most frequent words from them. The resulting vocabularies served as the input to our translation HIT.

Translation HIT For the translation task, we asked Turkers to translate individual words. We showed each word in the context of three sentences that were drawn from Wikipedia. Turkers were allowed to mark that they were unable to translate a word. Each task contained 10 words, 8 of which were words with unknown translations, and 2 of which were quality control words with known translations. We gave special instruction for translating names of people and places, giving examples of how to handle ‘Barack Obama’ and ‘Australia’ using their interlanguage links. For languages with non-Latin alphabets, names were transliterated.

The task paid \$0.15 for the translation of 10 words. Each set of 10 words was independently translated by three separate workers. 5,281 workers completed 256,604 translation assignments, totaling more than 3 million words, over a period of three and a half months.

Gold standard translations A set of gold standard translations were automatically harvested from

¹http://meta.wikimedia.org/wiki/List_of_Wikipedias

²<http://dumps.wikimedia.org/other/pagecounts-raw/>

500K+ ARTICLES: German (de), English (en), Spanish (es), French (fr), Italian (it), Japanese (ja), Dutch (nl), Polish (pl), Portuguese (pt), Russian (ru)
100K-500K ARTICLES: Arabic (ar), Bulgarian (bg), Catalan (ca), Czech (cs), Danish (da), Esperanto (eo), Basque (eu), Persian (fa), Finnish (fi), Hebrew (he), Hindi (hi), Croatian (hr), Hungarian (hu), Indonesian (id), Korean (ko), Lithuanian (lt), Malay (ms), Norwegian (Bokmal) (no), Romanian (ro), Slovak (sk), Slovenian (sl), Serbian (sr), Swedish (sv), Turkish (tr), Ukrainian (uk), Vietnamese (vi), Waray-Waray (war), Chinese (zh)
10K-100K ARTICLES: Afrikaans (af) Amharic (am) Asturian (ast) Azerbaijani (az) Belarusian (be) Bengali (bn) Bishnupriya Manipuri (bpy) Breton (br) Bosnian (bs) Cebuano (ceb) Welsh (cy) Zazaki (diq) Greek (el) West Frisian (fy) Irish (ga) Galician (gl) Gujarati (gu) Haitian (ht) Armenian (hy) Icelandic (is) Javanese (jv) Georgian (ka) Kannada (kn) Kurdish (ku) Luxembourgish (lb) Latvian (lv) Malagasy (mg) Macedonian (mk) Malayalam (ml) Marathi (mr) Neapolitan (nap) Low Saxon (nds) Nepali (ne) Newar / Nepal Bhasa (new) Norwegian (Nynorsk) (nn) Piedmontese (pms) Sicilian (scn) Serbo-Croatian (sh) Albanian (sq) Sundanese (su) Swahili (sw) Tamil (ta) Telugu (te) Thai (th) Tagalog (tl) Urdu (ur) Yoruba (yo)
< 10K ARTICLES: Central Bicolano (bcl) Tibetan (bo) Ilokano (ilo) Punjabi (pa) Kapampangan (pam) Pashto (ps) Sindhi (sd) Somali (so) Uzbek (uz) Wolof (wo)

Table 2: A list of the languages that were used in our study, grouped by the number of Wikipedia articles in the language. Each language’s code is given in parentheses. These language codes are used in other figures throughout this paper.

Wikipedia for every language to use as embedded controls. We used Wikipedia’s inter-language links to pair titles of English articles with their corresponding foreign article’s title. To get a more translatable set of pairs, we excluded any pairs where: (1) the English word was not present in the WordNet ontology (Miller, 1995), (2) either article title was longer than a single word, (3) the English Wikipedia page was a subcategory of person or place, or (4) the English and the foreign titles were identical or a substring of the other.

Manual evaluation of non-identical translations

We counted all translations that exactly matched the gold standard translation as correct. For non-exact matches we created a second-pass quality assurance HIT. Turkers were shown a pair of English words, one of which was a Turker’s translation of the foreign word used for quality control, and the other of which was the gold-standard translation of the foreign word. Evaluators were asked whether the two words had the same meaning, and chose between three answers: ‘Yes’, ‘No’, or ‘Re-

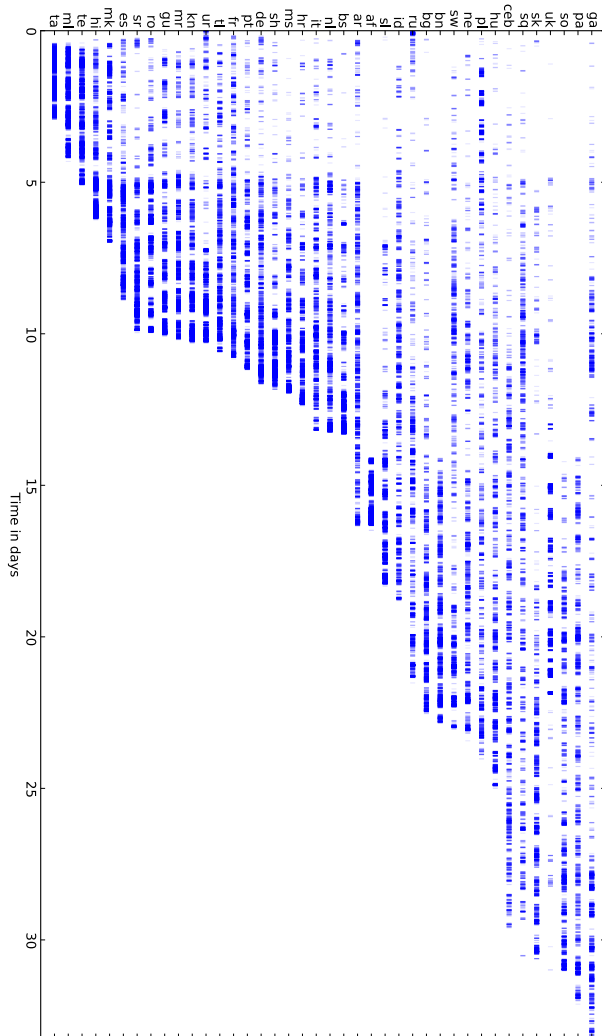


Figure 2: Days to complete the translation HITs for 40 of the languages. Tick marks represent the completion of individual assignments.

lated but not synonymous.’ Examples of meaning equivalent pairs include: *<petroglyphs, rock paintings>*, *<demo, show>* and *<loam, loam: soil rich in decaying matter>*. Non-meaning equivalents included: *<assorted, minutes>*, and *<major, URL of image>*. Related items were things like *<sky, clouds>*. Misspellings like *<lactation, lactation >* were judged to have same meaning, and were marked as misspelled. Three separate Turkers judged each pair, allowing majority votes for difficult cases.

We checked Turkers who were working on this task by embedding pairs of words which were ei-

पाकिस्तान ने भी प्रतिकार स्वरूप २८ मई १९९८ में छह परमाणु परीक्षण कर डाले

In retribution pakistan also did six nuclear tests on 28 may 1998. On 28 May Pakistan also conducted six nuclear tests as an act of redressal.

Retaliating on this 'Pakistan' conducted Six(6) Nuclear Tests on 28 May, 1998.

pakistan also did 6 nuclear test in retribution on 28 may, 1998

Figure 3: An example of the Turkers’ translations of a Hindi sentence. The translations are unedited and contain fixable spelling, capitalization and grammatical errors.

ther known to be synonyms (drawn from WordNet) or unrelated (randomly chosen from a corpus). Automating approval/rejections for the second-pass evaluation allowed the whole pipeline to be run automatically. Caching judgments meant that we ultimately needed only 20,952 synonym tasks to judge all of the submitted translations (a total of 74,572 non-matching word pairs). These were completed by an additional 1,005 workers. Each of these assignments included 10 word pairs and paid \$0.10.

Full sentence translations To demonstrate the feasibility of using crowdsourcing to create multilingual technologies, we hire Turkers to construct bilingual parallel corpora from scratch for six Indian languages. Germann (2001) attempted to build a Tamil-English translation system from scratch by hiring professional translators, but found the cost prohibitive. We created parallel corpora by translating the 100 most viewed Wikipedia pages in Bengali, Malyalam, Hindi, Tamil, Telugu, and Urdu into English. We collected four translations from different Turkers for each source sentence.

Workers were paid \$0.70 per HIT to translate 10 sentences. We accepted or rejected translations based on a manual review of each worker’s submissions, which included a comparison of the translations to a monotonic gloss (produced with a dictionary), and metadata such as the amount of time the worker took to complete the HIT and their geographic location.

Figure 3 shows an example of the translations we obtained. The lack of a professionally translated reference sentences prevented us from doing a systematic comparison between the quality of profes-

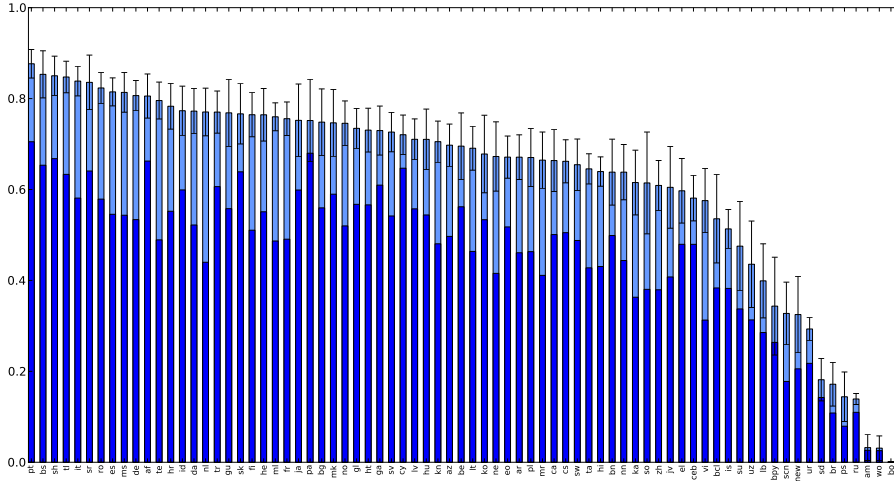


Figure 4: Translation quality for languages with at least 50 Turkers. The dark blue bars indicate the proportion of translations which exactly matched gold standard translations, and light blue indicate translations which were judged to be correct synonyms. Error bars show the 95% confidence intervals for each language.

sion and non-professional translations as Zaidan and Callison-Burch (2011) did. Instead we evaluate the quality of the data by using it to train SMT systems. We present results in section 5.

4 Measuring Translation Quality

For single word translations, we calculate the quality of translations on the level of individual assignments and aggregated over workers and languages. We define an assignment’s quality as the proportion of controls that are correct in a given assignment, where correct means exactly correct or judged to be synonymous.

$$\text{Quality}(a_i) = \frac{1}{k_i} \sum_{j=1}^{k_i} \delta(tr_{ij} \in \text{syns}[g_j]) \quad (1)$$

where a_i is the i^{th} assignment, k_i is the number of controls in a_i , tr_{ij} is the Turker’s provided translation of control word j in assignment i , g_j is the gold standard translation of control word j , $\text{syns}[g_j]$ is the set of words judged to be synonymous with g_j and includes g_j , and $\delta(x)$ is Kronecker’s delta and takes value 1 when x is true. Most assignments had two known words embedded, so most assignments had scores of either 0, 0.5, or 1.

Since computing overall quality for a language as the average assignment quality score is biased towards a small number of highly active Turkers, we instead report language quality scores as the average per-Turker quality, where a Turker’s quality is the average quality of all the assignments that she completed:

$$\text{Quality}(t_i) = \frac{\sum_{a_j \in \text{assigns}[i]} \text{Quality}(a_j)}{|\text{assigns}[i]|} \quad (2)$$

where $\text{assigns}[i]$ is the assignments completed by Turker i , and $\text{Quality}(a)$ is as above.

Quality for a language is then given by

$$\text{Quality}(l_i) = \frac{\sum_{t_j \in \text{turkers}[i]} \text{Quality}(t_j)}{|\text{turkers}[i]|} \quad (3)$$

When a Turker completed assignments in more than one language, their quality was computed separately for each language. Figure 4 shows the translation quality for languages with contributions from at least 50 workers.

Cheating using machine translation One obvious way for workers to cheat is to use available online translation tools. Although we followed best practices to deter copying-and-pasting into online MT systems by rendering words and sentences

as images (Zaidan and Callison-Burch, 2011), this strategy does not prevent workers from typing the words into an MT system if they are able to type in the language’s script.

To identify and remove workers who appeared to be cheating by using Google Translate, we calculated each worker’s overlap with the Google translations. We used Google to translate all 10,000 words for the 51 foreign languages that Google Translate covered at the time of the study. We measured the percent of workers’ translations that exactly matched the translation returned from Google.

Figure 5a shows overlap between Turkers’s translations and Google Translate. When overlap is high, it seems likely that those Turkers are cheating. It is also reasonable to assume that honest workers will overlap with Google some amount of the time as Google’s translations are usually accurate. We divide the workers into three groups: those with very high overlap with Google (likely cheating by using Google to translate words), those with reasonable overlap, and those with no overlap (likely cheating by other means, for instance, by submitting random text).

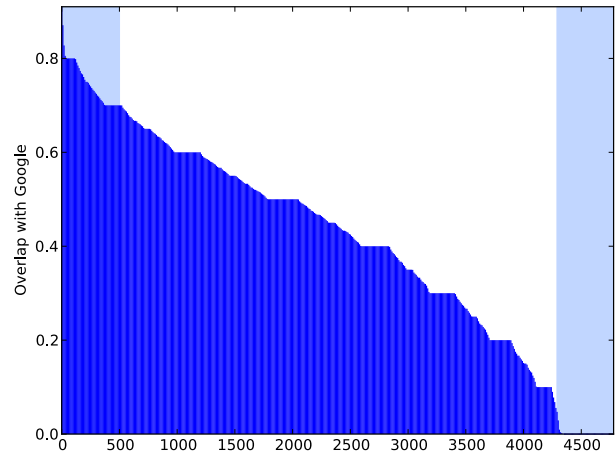
Our gold-standard controls are designed to identify workers that fall into the third group (those who are spamming or providing useless translations), but they will not effectively flag workers who are cheating with Google Translate. We therefore remove the 500 Turkers with the highest overlap with Google. This equates to removing all workers with greater than 70% overlap. Figure 5b shows that removing workers at or above the 70% threshold retains 90% of the collected translations and over 90% of the workers.

Quality scores reported throughout the paper reflect only translations from Turkers whose overlap with Google falls below this 70% threshold.

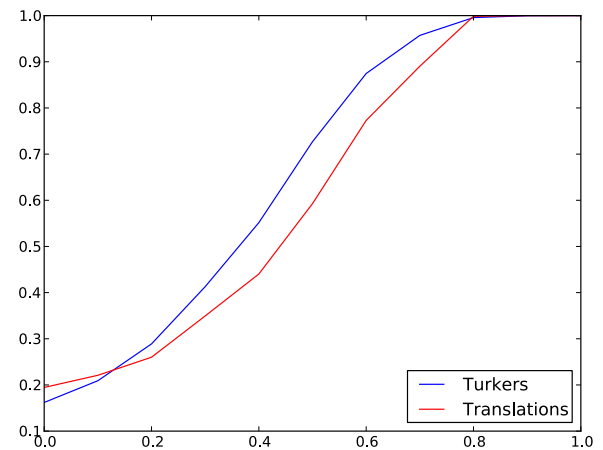
5 Data Analysis

We performed an analysis of our data to address the following questions:

- Do workers accurately represent their language abilities? Should we constrain tasks by region?
- How quickly can we expect work to be completed in a particular language?



(a) Individual workers’ overlap with Google Translate. We removed the 500 workers with the highest overlap (shaded region on the left) from our analyses, as it is reasonable to assume these workers are cheating by submitting translations from Google. Workers with no overlap (shaded region on the right) are also likely to be cheating, e.g. by submitting random text.



(b) Cumulative distribution of overlap with Google translate for workers and translations. We see that eliminating all workers with >70% overlap with google translate still preserves 90% of translations and >90% of workers.

Figure 5

- Can Turkers’ translations be used to train MT systems?
- Do our dictionaries improve MT quality?

Language skills and location We measured the average quality of workers who were in countries that plausibly speak a language, versus workers from countries that did not have large speaker populations of that language. We used the Ethnologue (Lewis

	Avg. Turker quality (# Ts)		Primary locations of Turkers in region	Primary locations of Turkers out of region
	In region	Out of region		
Hindi	0.63 (296)	0.69 (7)	India (284) UAE (5) UK (3)	Saudi Arabia (2) Russia (1) Oman (1)
Tamil	0.65 (273) **	0.25 (2)	India (266) US (3) Canada (2)	Tunisia (1) Egypt (1)
Malayalam	0.76 (234)	0.83 (2)	India (223) UAE (6) US (3)	Saudi Arabia (1) Maldives (1)
Spanish	0.81 (191)	0.84 (18)	US (122) Mexico (16) Spain (14)	India (15) New Zealand (1) Brazil (1)
French	0.75 (170)	0.82 (11)	India (62) US (45) France (23)	Greece (2) Netherlands (1) Japan (1)
Chinese	0.60 (116)	0.55 (21)	US (75) Singapore (13) China (9)	Hong Kong (6) Australia (3) Germany (2)
German	0.82 (91)	0.77 (41)	Germany (48) US (25) Austria (7)	India (34) Netherlands (1) Greece (1)
Italian	0.86 (90) *	0.80 (42)	Italy (42) US (29) Romania (7)	India (33) Ireland (2) Spain (2)
Amharic	0.14 (16) **	0.01 (99)	US (14) Ethiopia (2)	India (70) Georgia (9) Macedonia (5)
Kannada	0.70 (105)	NA (0)	India (105)	
Arabic	0.74 (60) **	0.60 (45)	Egypt (19) Jordan (16) Morocco (9)	US (19) India (11) Canada (3)
Sindhi	0.19 (96)	0.06 (9)	India (58) Pakistan (37) US (1)	Macedonia (4) Georgia (2) Indonesia (2)
Portuguese	0.87 (101)	0.96 (3)	Brazil (44) Portugal (31) US (15)	Romania (1) Japan (1) Israel (1)
Turkish	0.76 (76)	0.80 (27)	Turkey (38) US (18) Macedonia (8)	India (19) Pakistan (4) Taiwan (1)
Telugu	0.80 (102)	0.50 (1)	India (98) US (3) UAE (1)	Saudi Arabia (1)
Irish	0.74 (54)	0.71 (47)	US (39) Ireland (13) UK (2)	India (36) Romania (5) Macedonia (2)
Swedish	0.73 (54)	0.71 (45)	US (25) Sweden (22) Finland (3)	India (23) Macedonia (6) Croatia (2)
Czech	0.71 (45) *	0.61 (50)	US (17) Czech Republic (14) Serbia (5)	Macedonia (22) India (10) UK (5)
Russian	0.15 (67) *	0.12 (27)	US (36) Moldova (7) Russia (6)	India (14) Macedonia (4) UK (3)
Breton	0.17 (3)	0.18 (89)	US (3)	India (83) Macedonia (2) China (1)

Table 3: Translation quality when partitioning the translations into two groups, one containing translations submitted by Turkers whose location is within regions that plausibly speak the foreign language, and the other containing translations from Turkers outside those regions. In general, in-region Turkers provide higher quality translations. (**) indicates differences significant at $p=0.05$, (*) at $p=0.10$.

et al., 2013) to compile the list of countries where each language is spoken. Table 3 compares the average translation quality of assignments completed within the region of each language, and compares it to the quality of assignments completed outside that region.

Our workers reported speaking 95 languages natively. US workers alone reported 61 native languages. Overall, 4,297 workers were located in a region likely to speak the language from which they were translating, and 2,778 workers were located in countries considered out of region (meaning that about a third of our 5,281 Turkers completed HITs in multiple languages).

Table 3 shows the differences in translation quality when computed using in-region versus out-of-region Turkers, for the languages with the greatest number of workers. Within region workers typically produced higher quality translations. Given the number of Indian workers on Mechanical Turk, it is unsurprising that they represent majority of out-of-region workers. For the languages that had more than 75 out of region workers (Malay, Amharic, Icelandic, Sicilian, Wolof, and Breton), Indian workers represented at least 70% of the out of region workers

in each language.

A few languages stand out for having suspiciously strong performance by out of region workers, notably Irish and Swedish, for which out of region workers account for a near equivalent volume and quality of translations to the in region workers. This is admittedly implausible, considering the relatively small number of Irish speakers worldwide, and the very low number living in the countries in which our Turkers were based (primarily India). Such results highlight the fact that cheating using online translation resources is a real problem, and despite our best efforts to remove workers using Google Translate, some cheating is still evident. Restricting to within region workers is an effective way to reduce the prevalence of cheating. We discuss the languages which are best supported by true native speakers in section 6.

Speed of translation Figure 2 gives the completion times for 40 languages. The 10 languages to finish in the shortest amount of time were: Tamil, Malayalam, Telugu, Hindi, Macedonian, Spanish, Serbian, Romanian, Gujarati, and Marathi. Seven of the ten fastest languages are from India, which is un-

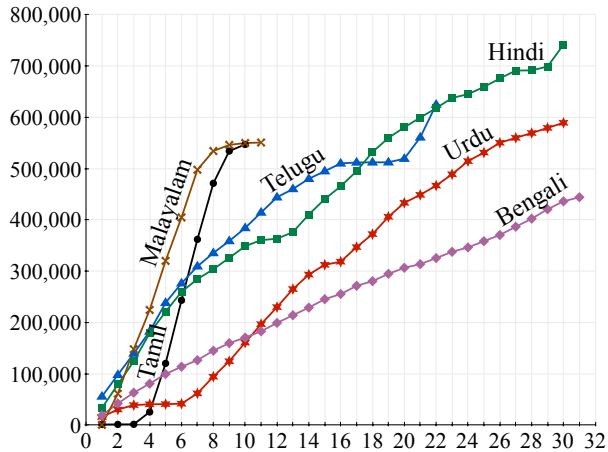


Figure 6: The total volume of translations (measured in English words) as a function of elapsed days.

language	sentence pairs	English + foreign words	dictionary entries
Bengali	22k	732k	22k
Hindi	40k	1,488k	22k
Malayalam	32k	863k	23k
Tamil	38k	916k	25k
Telugu	46k	1,097k	21k
Urdu	35k	1,356k	20k

Table 4: Size of parallel corpora and bilingual dictionaries collected for each language.

surprising given the geographic distribution of workers. Some languages follow the pattern of having a smattering of assignments completed early, with the rate picking up later.

Figure 6 gives the throughput of the full-sentence translation task for the six Indian languages. The fastest language was Malayalam, for which we collected half a million words of translations in just under a week. Table 4 gives the size of the data set that we created for each of these languages.

Training SMT systems We trained statistical translation models from the parallel corpora that we created for the six Indian languages using the Joshua machine translation system (Post et al., 2012). Table 5 shows the translation performance when trained on the bitexts alone, and when incorporating the bilingual dictionaries created in our earlier HIT. The scores reflect the performance when tested on held out sentences from the training data. Adding the dic-

language	trained on bitexts alone	bitext + dictionaries	BLEU Δ
Bengali	12.03	17.29	5.26
Hindi	16.19	18.10	1.91
Malayalam	6.65	9.72	3.07
Tamil	8.08	9.66	1.58
Telugu	11.94	13.70	1.76
Urdu	19.22	21.98	2.76

Table 5: BLEU scores for translating into English using bilingual parallel corpora by themselves, and with the addition of single-word dictionaries. Scores are calculated using four reference translations and represent the mean of three MERT runs.

tionaries to the training set produces consistent performance gains, ranging from 1 to 5 BLEU points. This represents a substantial improvement. It is worth noting, however, that while the source documents for the full sentences used for testing were kept disjoint from those used for training, there is overlap between the source materials for the dictionaries and those from the test set, since both the dictionaries and the bitext source sentences were drawn from Wikipedia.

6 Discussion

Crowdsourcing platforms like Mechanical Turk give researchers instant access to a diverse set of bilingual workers. This opens up exciting new avenues for researchers to develop new multilingual systems. The demographics reported in this study are likely to shift over time. Amazon may expand its payments to new currencies. Posting long-running HITs in other languages may recruit more speakers of those languages. New crowdsourcing platforms may emerge. The data presented here provides a valuable snapshot of the current state of MTurk, and the methods used can be applied generally in future research.

Based on our study, we can confidently recommend 13 languages as good candidates for research now: Dutch, French, German, Gujarati, Italian, Kananda, Malayalam, Portuguese, Romanian, Serbian, Spanish, Tagalog, and Telugu. These languages have large Turker populations who complete tasks quickly and accurately. Table 6 summarizes the strengths and weaknesses of all 100 languages covered in our study. Several other languages are viable

workers	quality	speed	
many	high	fast	Dutch, French, German, Gujarati, Italian, Kannada, Malayalam, Portuguese, Romanian, Serbian, Spanish, Tagalog, Telugu
		slow	Arabic, Hebrew, Irish, Punjabi, Swedish, Turkish
	low or medium	fast	Hindi, Marathi, Tamil, Urdu
		slow	Bengali, Bishnupriya Manipuri, Cebuano, Chinese, Nepali, Newar, Polish, Russian, Sindhi, Tibetan
few	high	fast	Bosnia, Croatian, Macedonian, Malay, Serbo-Croatian
		slow	Afrikaans, Albanian, Aragonese, Asturian, Basque, Belarusian, Bulgarian, Central Bicolano, Czech, Danish, Finnish, Galacian, Greek, Haitian, Hungarian, Icelandic, Ilokano, Indonesian, Japanese, Javanese, Kapampangan, Kazakh, Korean, Lithuanian, Low Saxon, Malagasy, Norwegian (Bokmal), Sicilian, Slovak, Slovenian, Thai, Ukrainian, Uzbek, Waray-Waray, West Frisian, Yoruba
	low or medium	fast	–
		slow	Amharic, Armenian, Azerbaijani, Breton, Catalan, Georgian, Latvian, Luxembourgish, Neapolitan, Norwegian (Nynorsk), Pashto, Piedmontese, Somali, Sudanese, Swahili, Tatar, Vietnamese, Walloon, Welsh
none	low or medium	slow	Esperanto, Ido, Kurdish, Persian, Quechua, Wolof, Zazaki

Table 6: The green box shows the best languages to target on MTurk. These languages have many workers who generate high quality results quickly. We defined *many* workers as 50 or more active in-region workers, *high* quality as $\geq 70\%$ accuracy on the gold standard controls, and *fast* if all of the 10,000 words were completed within two weeks.

candidates provided adequate quality control mechanisms are used to select good workers.

Since Mechanical Turk provides financial incentives for participation, many workers attempt to complete tasks even if they do not have the language skills necessary to do so. Since MTurk does not provide any information about workers demographics, including their language competencies, it can be hard to exclude such workers. As a result naive data collection on MTurk may result in noisy data. A variety of techniques should be incorporated into crowdsourcing pipelines to ensure high quality data. As a best practice, we suggest: (1) restricting workers to countries that plausibly speak the foreign language of interest, (2) embedding gold standard controls or administering language pretests, rather than relying solely on self-reported language skills, and (3) excluding workers whose translations have high overlap with online machine translation systems like Google translate. If cheating using external resources is likely, then also consider (4) recording information like time spent on a HIT (cumulative and on individual items), patterns in keystroke logs, tab/window focus, etc.

Although our study targeted bilingual workers on Mechanical Turk, and neglected monolingual workers, we believe our results reliably represent the current speaker populations, since the vast majority of the work available on the crowdsourced platform is currently English-only. We therefore assume the number of non-English speakers is small. In the future, it may be desirable to recruit monolingual foreign workers. In such cases, we recommend other tests to validate their language abilities in place of our translation test. These could include performing narrative cloze, or listening to audio files containing speech in different language and identifying their language.

7 Data release

With the publication of this paper, we are releasing all data and code used in this study. Our data release includes the raw data, along with bilingual dictionaries that are filtered to be high quality. It will include 256,604 translation assignments from 5,281 Turkers and 20,952 synonym assignments from 1,005 Turkers, along with meta information like geolocation

and time submitted, plus external dictionaries used for validation. The dictionaries will contain 1.5M total translated words in 100 languages, along with code to filter the dictionaries based on different criteria. The data also includes parallel corpora for six Indian languages, ranging in size between 700,000 to 1.5 million words.

8 Acknowledgements

This material is based on research sponsored by a DARPA Computer Science Study Panel phase 3 award entitled “Crowdsourcing Translation” (contract D12PC00368). The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements by DARPA or the U.S. Government. This research was supported by the Johns Hopkins University Human Language Technology Center of Excellence and through gifts from Microsoft and Google.

The authors would like to thank the anonymous reviewers for their thoughtful comments, which substantially improved this paper.

References

- Amazon. 2013. Service summary tour for requesters on Amazon Mechanical Turk. <https://requester.mturk.com/tour>.
- Vamshi Ambati and Stephan Vogel. 2010. Can crowds build parallel corpora for machine translation systems? In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon’s Mechanical Turk*. Association for Computational Linguistics.
- Vamshi Ambati, Stephan Vogel, and Jaime Carbonell. 2010. Active learning and crowd-sourcing for machine translation. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC)*.
- Vamshi Ambati. 2012. *Active Learning and Crowdsourcing for Machine Translation in Low Resource Scenarios*. Ph.D. thesis, Language Technologies Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.
- Michael S. Bernstein, Greg Little, Robert C. Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. 2010. Soylent: a word processor with a crowd inside. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*.
- Jeffrey P. Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C. Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, Samuel White, and Tom Yeh. 2010. VizWiz: nearly real-time answers to visual questions. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*.
- Michael Bloodgood and Chris Callison-Burch. 2010. Large-scale cost-focused active learning for statistical machine translation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.
- Chris Callison-Burch and Mark Dredze. 2010. Creating speech and language data with Amazon’s Mechanical Turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon’s Mechanical Turk*, pages 1–12, Los Angeles, June. Association for Computational Linguistics.
- Jia Deng, Alexander Berg, Kai Li, and Li Fei-Fei. 2010. What does classifying more than 10,000 image categories tell us? In *Proceedings of the 12th European Conference of Computer Vision (ECCV)*, pages 71–84.
- Maxine Eskenazi, Gina-Anne Levow, Helen Meng, Gabriel Parent, and David Suendermann. 2013. *Crowdsourcing for Speech Processing, Applications to Data Collection, Transcription and Assessment*. Wiley.
- Siamak Faridani, Björn Hartmann, and Panagiotis G. Ipeirotis. 2011. What’s the right price? pricing tasks for finishing on time. In *Third AAAI Human Computation Workshop (HCOMP’11)*.
- Ulrich Germann. 2001. Building a statistical machine translation system from scratch: How much bang for the buck can we expect? In *ACL 2001 Workshop on Data-Driven Machine Translation*, Toulouse, France.
- Chang Hu, Benjamin B. Bederson, and Philip Resnik. 2010. Translation by iterative collaboration between monolingual users. In *Proceedings of ACM SIGKDD Workshop on Human Computation (HCOMP)*.
- Chang Hu, Philip Resnik, Yakov Kronrod, Vladimir Eidelman, Olivia Buzek, and Benjamin B. Bederson. 2011. The value of monolingual crowdsourcing in a real-world translation scenario: Simulation using haitian creole emergency sms messages. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 399–404, Edinburgh, Scotland, July. Association for Computational Linguistics.
- Panagiotis G. Ipeirotis. 2010a. Analyzing the mechanical turk marketplace. In *ACM XRDS*, December.
- Panagiotis G. Ipeirotis. 2010b. Demographics of Mechanical Turk. Technical Report Working paper

- CeDER-10-01, New York University, Stern School of Business.
- Ann Irvine and Alexandre Klementiev. 2010. Using Mechanical Turk to annotate lexicons for less commonly used languages. In *Workshop on Creating Speech and Language Data with MTurk*.
- Ian Lane, Matthias Eck, Kay Rottmann, and Alex Waibel. 2010. Tools for collecting speech corpora via mechanical-turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, Los Angeles.
- Florian Laws, Christian Scheible, and Hinrich Schütze. 2011. Active learning with amazon mechanical turk. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, Edinburgh, Scotland.
- Matthew Lease, Jessica Hullman, Jeffrey P. Bigham, Juho Kim Michael S. Bernstein and, Walter Lasecki, Saeideh Bakhshi, Tanushree Mitra, and Robert C. Miller. 2013. Mechanical Turk is not anonymous. <http://dx.doi.org/10.2139/ssrn.2228728>.
- Vili Lehdonvirta and Mirko Ernkvist. 2011. Knowledge map of the virtual economy: Converting the virtual economy into development potential. <http://www.infodev.org/en/Document.1056.pdf>, April. An InfoDev Publication.
- M. Paul Lewis, Gary F. Simons, and Charles D. Fennig (eds.). 2013. *Ethnologue: Languages of the world*, seventeenth edition. <http://www.ethnologue.com>.
- Greg Little, Lydia B. Chilton, Rob Miller, and Max Goldman. 2009. TurkIt: Tools for iterative tasks on mechanical turk. In *Proceedings of the Workshop on Human Computation at the International Conference on Knowledge Discovery and Data Mining (KDD-HCOMP '09)*, Paris.
- Matthew Marge, Satanjeev Banerjee, and Alexander Rudnicky. 2010. Using the Amazon Mechanical Turk to transcribe and annotate meeting speech for extractive summarization. In *Workshop on Creating Speech and Language Data with MTurk*.
- George A. Miller. 1995. WordNet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.
- Robert Munro and Hal Tily. 2011. The start of the art: Introduction to the workshop on crowdsourcing technologies for language and cognition studies. In *Crowdsourcing Technologies for Language and Cognition Studies*, Boulder.
- Scott Novotney and Chris Callison-Burch. 2010. Cheap, fast and good enough: Automatic speech recognition with non-expert transcription. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 207–215. Association for Computational Linguistics.
- Gabriel Parent and Maxine Eskenazi. 2011. Speaking to the crowd: looking at past achievements in using crowdsourcing for speech and predicting future challenges. In *Proceedings Interspeech 2011, Special Session on Crowdsourcing*.
- Matt Post, Chris Callison-Burch, and Miles Osborne. 2012. Constructing parallel corpora for six indian languages via crowdsourcing. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 401–409, Montréal, Canada, June. Association for Computational Linguistics.
- Alexander J. Quinn and Benjamin B. Bederson. 2011. Human computation: A survey and taxonomy of a growing field. In *Computer Human Interaction (CHI)*.
- Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. 2010. Collecting image annotations using Amazon's Mechanical Turk. In *Workshop on Creating Speech and Language Data with MTurk*.
- Joel Ross, Lilly Irani, M. Six Silberman, Andrew Zaldivar, and Bill Tomlinson. 2010. Who are the crowdworkers?: Shifting demographics in Amazon Mechanical Turk. In *alt.CHI session of CHI 2010 extended abstracts on human factors in computing systems*, Atlanta, Georgia.
- Yaron Singer and Manas Mittal. 2011. Pricing mechanisms for online labor markets. In *Third AAAI Human Computation Workshop (HCOMP'11)*.
- Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast - but is it good? Evaluating non-expert annotations for natural language tasks. In *Proceedings of EMNLP*.
- Alexander Sorokin and David Forsyth. 2008. Utility data annotation with amazon mechanical turk. In *First IEEE Workshop on Internet Vision at CVPR*.
- Luis von Ahn. 2005. *Human Computation*. Ph.D. thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.
- Omar F. Zaidan and Chris Callison-Burch. 2011. Crowdsourcing translation: Professional quality from non-professionals. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1220–1229. Association for Computational Linguistics.
- Rabih Zbib, Erika Malchiodi, Jacob Devlin, David Stallard, Spyros Matsoukas, Richard Schwartz, John Makhoul, Omar F. Zaidan, and Chris Callison-Burch. 2012. Machine translation of Arabic dialects. In *The 2012 Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Rabih Zbib, Gretchen Markiewicz, Spyros Matsoukas, Richard Schwartz, and John Makhoul. 2013. Systematic comparison of professional and crowdsourced reference translations for machine translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Atlanta, Georgia.

Cross-lingual Projected Expectation Regularization for Weakly Supervised Learning

Mengqiu Wang and Christopher D. Manning

Computer Science Department

Stanford University

Stanford, CA 94305 USA

{mengqiu, manning}@cs.stanford.edu

Abstract

We consider a multilingual weakly supervised learning scenario where knowledge from annotated corpora in a resource-rich language is transferred via bitext to guide the learning in other languages. Past approaches project labels across bitext and use them as features or gold labels for training. We propose a new method that projects model expectations rather than labels, which facilitates transfer of model uncertainty across language boundaries. We encode expectations as constraints and train a discriminative CRF model using Generalized Expectation Criteria (Mann and McCallum, 2010). Evaluated on standard Chinese-English and German-English NER datasets, our method demonstrates F_1 scores of 64% and 60% when no labeled data is used. Attaining the same accuracy with supervised CRFs requires 12k and 1.5k labeled sentences. Furthermore, when combined with labeled examples, our method yields significant improvements over state-of-the-art supervised methods, achieving best reported numbers to date on Chinese OntoNotes and German CoNLL-03 datasets.

1 Introduction

Supervised statistical learning methods have enjoyed great popularity in Natural Language Processing (NLP) over the past decade. The success of supervised methods depends heavily upon the availability of large amounts of annotated training data. Manual curation of annotated corpora is a costly and time consuming process. To date, most annotated resources reside within the English language, which

hinders the adoption of supervised learning methods in many multilingual environments.

To minimize the need for annotation, significant progress has been made in developing unsupervised and semi-supervised approaches to NLP (Collins and Singer 1999; Klein 2005; Liang 2005; Smith 2006; Goldberg 2010; *inter alia*). More recent paradigms for semi-supervised learning allow modelers to directly encode knowledge about the task and the domain as constraints to guide learning (Chang et al., 2007; Mann and McCallum, 2010; Ganchev et al., 2010). However, in a multilingual setting, coming up with effective constraints require extensive knowledge of the foreign¹ language.

Bilingual parallel text (bitext) lends itself as a medium to transfer knowledge from a resource-rich language to a foreign language. Yarowsky and Ngai (2001) project labels produced by an English tagger to the foreign side of bitext, then use the projected labels to learn a HMM model. More recent work applied the projection-based approach to more language-pairs, and further improved performance through the use of type-level constraints from tag dictionary and feature-rich generative or discriminative models (Das and Petrov, 2011; Täckström et al., 2013).

In our work, we propose a new projection-based method that differs in two important ways. First, we never explicitly project the labels. Instead, we project expectations over the labels. This projection

¹For experimental purposes, we designate English as the resource-rich language, and other languages of interest as “foreign”. In our experiments, we simulate the resource-poor scenario using Chinese and German, even though in reality these two languages are quite rich in resources.

acts as a soft constraint over the labels, which allows us to transfer more information and uncertainty across language boundaries. Secondly, we encode the expectations as constraints and train a model by minimizing divergence between model expectations and projected expectations in a Generalized Expectation (GE) Criteria (Mann and McCallum, 2010) framework.

We evaluate our approach on Named Entity Recognition (NER) tasks for English-Chinese and English-German language pairs on standard public datasets. We report results in two settings: a weakly supervised setting where no labeled data or a small amount of labeled data is available, and a semi-supervised settings where labeled data is available, but we can gain predictive power by learning from unlabeled bitext.

2 Related Work

Most semi-supervised learning approaches embody the principle of learning from constraints. There are two broad categories of constraints: multi-view constraints, and external knowledge constraints.

Examples of methods that explore multi-view constraints include self-training (Yarowsky, 1995; McClosky et al., 2006),² co-training (Blum and Mitchell, 1998; Sindhwani et al., 2005), multi-view learning (Ando and Zhang, 2005; Carlson et al., 2010), and discriminative and generative model combination (Suzuki and Isozaki, 2008; Druck and McCallum, 2010).

An early example of using knowledge as constraints in weakly-supervised learning is the work by Collins and Singer (1999). They showed that the addition of a small set of “seed” rules greatly improve a co-training style unsupervised tagger. Chang et al. (2007) proposed a constraint-driven learning (CODL) framework where constraints are used to guide the selection of best self-labeled examples to be included as additional training data in an iterative EM-style procedure. The kind of constraints used in applications such as NER are the ones like “the words CA, Australia, NY are LOCATION” (Chang et al., 2007). Notice the similarity of this partic-

²A multi-view interpretation of self-training is that the self-tagged additional data offers new views to learners trained on existing labeled data.

ular constraint to the kinds of features one would expect to see in a discriminative MaxEnt model. The difference is that instead of learning the validity (or weight) of this feature from labeled examples — since we do not have them — we can constrain the model using our knowledge of the domain. Druck et al. (2009) also demonstrated that in an active learning setting where annotation budget is limited, it is more efficient to label features than examples. Other sources of knowledge include lexicons and gazetteers (Druck et al., 2007; Chang et al., 2007).

While it is straight-forward to see how resources such as a list of city names can give a lot of mileage in recognizing locations, we are also exposed to the danger of over-committing to hard constraints. For example, it becomes problematic with city names that are ambiguous, such as Augusta, Georgia.³ To soften these constraints, Mann and McCallum (2010) proposed the Generalized Expectation (GE) Criteria framework, which encodes constraints as a regularization term over some score function that measures the divergence between the model’s expectation and the target expectation. The connection between GE and CODL is analogous to the relationship between hard (Viterbi) EM and soft EM, as illustrated by Samdani et al. (2012).

Another closely related work is the Posterior Regularization (PR) framework by Ganchev et al. (2010). In fact, as Bellare et al. (2009) have shown, in a discriminative model these two methods optimize exactly the same objective.⁴ The two differ in optimization details: PR uses a EM algorithm to approximate the gradients which avoids the expensive computation of a covariance matrix between features and constraints, whereas GE directly calculates the gradient. However, later results (Druck, 2011) have shown that using the Expectation Semiring techniques of Li and Eisner (2009), one can compute the exact gradients of GE in a Conditional Random Fields (CRF) (Lafferty et al., 2001) at costs

³This is a city in the state of Georgia in USA, famous for its golf courses. It is ambiguous since both Augusta and Georgia can also be used as person names.

⁴The different terminology employed by GE and PR may be confusing to discerning readers, but the “expectation” in the context of GE means the same thing as “marginal posterior” as in PR.

no greater than computing the gradients of ordinary CRF. And empirically, GE tends to perform more accurately than PR (Bellare et al., 2009; Druck, 2011).

Obtaining appropriate knowledge resources for constructing constraints remain as a bottleneck in applying GE and PR to new languages. However, a number of past work recognizes parallel bitext as a rich source of linguistic constraints, naturally captured in the translations. As a result, bitext has been effectively utilized for unsupervised multilingual grammar induction (Alshawi et al., 2000; Snyder et al., 2009), parsing (Burkett and Klein, 2008), and sequence labeling (Naseem et al., 2009).

A number of recent work also explored bilingual constraints in the context of simultaneous bilingual tagging, and showed that enforcing agreements between language pairs give superior results than monolingual tagging (Burkett et al., 2010; Che et al., 2013; Wang et al., 2013a). Burkett et al. (2010) also demonstrated a *uptraining* (Petrov et al., 2010) setting where tag-induced bitext can be used as additional monolingual training data to improve monolingual taggers. A major drawback of this approach is that it requires a readily-trained tagging models in each languages, which makes a weakly supervised setting infeasible. Another intricacy of this approach is that it only works when the two models have comparable strength, since mutual agreements are enforced between them.

Projection-based methods can be very effective in weakly-supervised scenarios, as demonstrated by Yarowsky and Ngai (2001), and Xi and Hwa (2005). One problem with projected labels is that they are often too noisy to be directly used as training signals. To mitigate this problem, Das and Petrov (2011) designed a label propagation method to automatically induce a tag lexicon for the foreign language to smooth the projected labels. Fossum and Abney (2005) filter out projection noise by combining projections from from multiple source languages. However, this approach is not always viable since it relies on having parallel bitext from multiple source languages. Li et al. (2012) proposed the use of crowd-sourced Wiktionary as additional resources for inducing tag lexicons. More recently, Täckström et al. (2013) combined token-level and type-level constraints to constrain legitimate label sequences and and recalibrate the probability distri-

bution in a CRF. The tag dictionary used for POS tagging are analogous to the gazetteers and name lexicons used for NER by Chang et al. (2007).

Our work is also closely related to Ganchev et al. (2009). They used a two-step projection method similar to Das and Petrov (2011) for dependency parsing. Instead of using the projected linguistic structures as ground truth (Yarowsky and Ngai, 2001), or as features in a generative model (Das and Petrov, 2011), they used them as constraints in a PR framework. Our work differs by projecting expectations rather than Viterbi one-best labels. We also choose the GE framework over PR. Experiments in Bellare et al. (2009) and Druck (2011) suggest that in a discriminative model (like ours), GE is more accurate than PR. More recently, Ganchev and Das (2013) further extended this line of work to directly train discriminative sequence models using cross lingual projection with PR. The types of constraints applied in this new work are similar to the ones in the monolingual PR setting proposed by Ganchev et al. (2010), where the total counts of labels of a particular kind are expected to match some fraction of the projected total counts. Our work differ in that we enforce expectation constraints at token level, which gives tighter guidance to learning the model.

3 Approach

Given bitext between English and a foreign language, our goal is to learn a CRF model in the foreign language from little or no labeled data. Our method performs **Cross-Lingual Projected Expectation Regularization (CLiPER)**.

For every aligned sentence pair in the bitext, we first compute the posterior marginal at each word position on the English side using a pre-trained English CRF tagger; then for each aligned English word, we project its posterior marginal as expectations to the aligned word position on the foreign side. Figure 1 shows a snippet of a sentence from real corpus. Notice that if we were to directly project the Viterbi best assignment from English to Chinese, all three Chinese words that are named entities would have gotten the wrong tags. But projecting the English CRF model expectations preserves some uncertainties, informing the Chinese model that there is a 40%

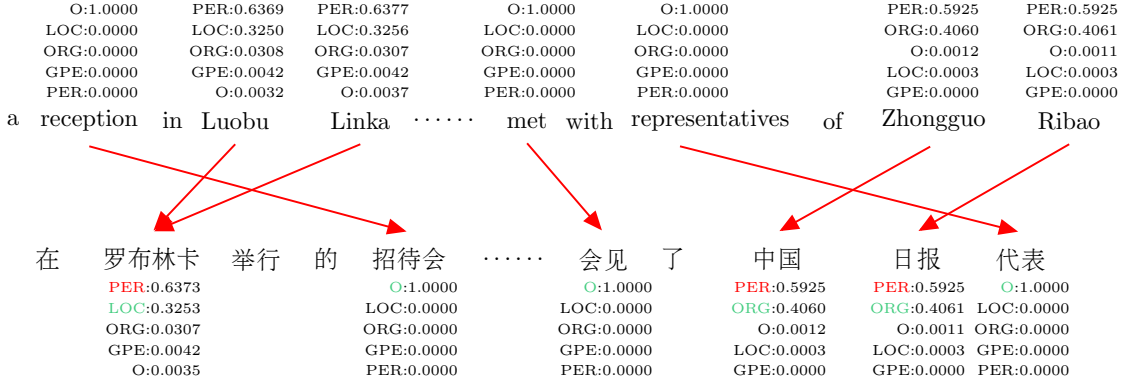


Figure 1: Diagram illustrating the projection of model expectation from English to Chinese. The posterior probabilities assigned by the English CRF model is shown above each English word; automatically induced word alignments are shown in red; the correct projected labels for Chinese words are shown in green, and incorrect labels are shown in red.

chance that “中国日报” (*China Daily*) is an organization in this context.

We would like to learn a CRF model in the foreign language that has similar expectations as the projected expectations from English. To this end, we adopt the Generalized Expectation (GE) Criteria framework introduced by Mann and McCallum (2010). In the remainder of this section, we follow the notation used in (Druck, 2011) to explain our approach.

3.1 CLiPER

The general idea of GE is that we can express our preferences over models through constraint functions. A desired model should satisfy the imposed constraints by matching the expectations on these constraint functions with some target expectations (attained by external knowledge like lexicons or in our case transferred knowledge from English). We define a constraint function ϕ_{i,l_j} for each word position i and output label assignment l_j . $\phi_{i,l_j} = 0$ is a constraint in that position i cannot take label l_j .

The set $\{l_1, \dots, l_m\}$ denotes all possible label assignment for each y_i , and m is number of label values. A_i is the set of English words aligned to Chinese word i . ϕ_{i,l_j} are defined for all position i such that $A_i \neq \emptyset$. In other words, the constraint function applies only to Chinese word positions that have at least one aligned English word. Each $\phi_{i,l_j}(\mathbf{y})$ can be treated as a Bernoulli random variable, and we concatenate the set of all ϕ_{i,l_j} into a *random vector*

$\phi(\mathbf{y})$, where $\phi_k = \phi_{i,l_j}$ if $k = i * m + j$. We drop the (\mathbf{y}) in ϕ for simplicity.

The target expectation over ϕ_{i,l_j} , denoted as $\tilde{\phi}_{i,l_j}$, is the expectation of assigning label l_j to English word A_i under the English conditional probability model. When multiple English words are aligned to the same foreign word, we average the expectations.

The expectation over ϕ under a conditional probability model $P(\mathbf{y}|\mathbf{x}; \theta)$ is denoted as $E_{P(\mathbf{y}|\mathbf{x}; \theta)}[\phi]$, and simplified as $E_\theta[\phi]$ whenever it is unambiguous.

The conditional probability model $P(\mathbf{y}|\mathbf{x}; \theta)$ in our case is defined as a standard linear-chain CRF:⁵

$$P(\mathbf{y}|\mathbf{x}; \theta) = \frac{1}{Z(\mathbf{x}; \theta)} \exp \left(\sum_i^n \theta \mathbf{f}(\mathbf{x}, y_i, y_{i-1}) \right)$$

where \mathbf{f} is a set of feature functions; θ are the matching parameters to learn; $n = |\mathbf{x}|$.

The objective function to maximize in a standard CRF is the log probability over a collection of labeled documents:

$$L_{CRF}(\theta) = \sum_{a=1}^{a'} \log P(\mathbf{y}_a^* | \mathbf{x}_a; \theta) \quad (1)$$

a' is the number of labeled sentences. \mathbf{y}^* is an observed label sequence.

The objective function to maximize in GE is defined as the sum over all unlabeled examples on the

⁵We simplify notation by dropping the L_2 regularizer in the CRF definition, but apply it in our experiments.

foreign side of bitext, denoted as \mathbf{x}_b , over some cost function S between the model expectation over ϕ ($E_\theta[\phi]$) and the target expectation ($\tilde{\phi}$).

We choose S to be the negative L_2^2 squared error sum⁶ defined as:

$$\begin{aligned} L_{GE}(\theta) &= \sum_{b=1}^{n'} S \left(E_{P(\mathbf{y}_b|\mathbf{x}_b;\theta)}[\phi(\mathbf{y}_b)], \tilde{\phi}_b \right) \\ &= \sum_{b=1}^{n'} -\|\tilde{\phi}_b - E_\theta[\phi(\mathbf{y}_b)]\|_2^2 \end{aligned} \quad (2)$$

n' is the total number of unlabeled bitext sentence pairs.

When both labeled and bitext training data are available, the joint objective is the sum of Eqn. 1 and 2. Each is computed over the labeled training data and foreign half in the bitext, respectively.

We can optimize this joint objective by computing the gradients and use a gradient-based optimization method such as L-BFGS. Gradients of L_{CRF} decomposes down to the gradients over each labeled training example $(\mathbf{x}, \mathbf{y}^*)$. Computing the gradient of L_{GE} decomposes down to the gradients of $S(E_{P(\mathbf{y}|\mathbf{x};\theta)}[\phi])$ for each unlabeled foreign sentence \mathbf{x} and the constraints over this example ϕ . The gradients can be calculated as:

$$\begin{aligned} \frac{\partial}{\partial \theta} S(E_\theta[\phi]) &= -\frac{\partial}{\partial \theta} \left(\tilde{\phi} - E_\theta[\phi] \right)^T \left(\tilde{\phi} - E_\theta[\phi] \right) \\ &= 2 \left(\tilde{\phi} - E_\theta[\phi] \right)^T \left(\frac{\partial}{\partial \theta} E_\theta[\phi] \right) \end{aligned}$$

We redefine the penalty vector $\mathbf{u} = 2 \left(\tilde{\phi} - E_\theta[\phi] \right)$ to be u . $\frac{\partial}{\partial \theta} E_\theta[\phi]$ is a matrix where each column contains the gradients for a particular model feature θ with respect to all constraint functions ϕ . It can be

⁶In general, other loss functions such as KL-divergence can also be used for S . We found L_2^2 to work well in practice.

computed as:

$$\begin{aligned} \frac{\partial}{\partial \theta} E_\theta[\phi] &= \sum_{\mathbf{y}} \phi(\mathbf{y}) \frac{\partial}{\partial \theta} P(\mathbf{y}|\mathbf{x}; \theta) \\ &= \sum_{\mathbf{y}} \phi(\mathbf{y}) \frac{\partial}{\partial \theta} \left(\frac{1}{Z(\mathbf{x}; \theta)} \exp(\theta^T \mathbf{f}(\mathbf{x}, \mathbf{y})) \right) \\ &= \sum_{\mathbf{y}} \phi(\mathbf{y}) \left(\frac{1}{Z(\mathbf{x}; \theta)} \left(\frac{\partial}{\partial \theta} \exp(\theta^T \mathbf{f}(\mathbf{x}, \mathbf{y})) \right) \right. \\ &\quad \left. + \exp(\theta^T \mathbf{f}(\mathbf{x}, \mathbf{y})) \left(\frac{\partial}{\partial \theta} \frac{1}{Z(\mathbf{x}; \theta)} \right) \right) \\ &= \sum_{\mathbf{y}} \phi(\mathbf{y}) \left(P(\mathbf{y}|\mathbf{x}; \theta) \mathbf{f}(\mathbf{x}, \mathbf{y})^T \right. \\ &\quad \left. - P(\mathbf{y}|\mathbf{x}; \theta) \sum_{\mathbf{y}'} P(\mathbf{y}'|\mathbf{x}; \theta) \mathbf{f}(\mathbf{x}, \mathbf{y}')^T \right) \\ &= \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}; \theta) \sum_{\mathbf{y}} \phi(\mathbf{y}) \mathbf{f}(\mathbf{x}, \mathbf{y})^T \\ &\quad - \left(\sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}; \theta) \phi(\mathbf{y}) \right) \left(\sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}; \theta) \mathbf{f}(\mathbf{x}, \mathbf{y})^T \right) \\ &= \text{COV}_{P(\mathbf{y}|\mathbf{x};\theta)}(\phi(\mathbf{y}), \mathbf{f}(\mathbf{x}, \mathbf{y})) \quad (3) \\ &= E_\theta[\phi \mathbf{f}^T] - E_\theta[\phi] E_\theta[\mathbf{f}^T] \quad (4) \end{aligned}$$

Eqn. 3 gives the intuition of how optimization works in GE. In each iteration of L-BFGS, the model parameters are updated according to their covariance with the constraint features, scaled by the difference between current expectation and target expectation. The term $E_\theta[\phi \mathbf{f}^T]$ in Eqn. 4 can be computed using a dynamic programming (DP) algorithm, but solving it directly requires us to store a matrix of the same dimension as \mathbf{f}^T in each step of the DP. We can reduce the complexity by using the same trick as in (Li and Eisner, 2009) for computing Expectation Semiring. The resulting algorithm has complexity $O(nm^2)$, which is the same as the standard forward-backward inference algorithm for CRF. (Druck, 2011, 93) gives full details of this derivation.

3.2 Hard vs. soft Projection

Projecting expectations instead of one-best label assignments from English to foreign language can be thought of as a soft version of the method described in (Das and Petrov, 2011) and (Ganchev et

al., 2009). Soft projection has its advantage: when the English model is not certain about its predictions, we do not have to commit to the current best prediction. The foreign model has more freedom to form its own belief since any marginal distribution it produces would deviate from a flat distribution by just about the same amount. In general, preserving uncertainties till later is a strategy that has benefited many NLP tasks (Finkel et al., 2006). Hard projection can also be treated as a special case in our framework. We can simply recalibrate posterior marginal of English by assigning probability mass 1 to the most likely outcome, and zero everything else out, effectively taking the argmax of the marginal at each word position. We refer to this version of expectation as the “hard” expectation. In the hard projection setting, GE training resembles a “project-then-train” style semi-supervised CRF training scheme (Yarowsky and Ngai, 2001; Täckström et al., 2013). In such a training scheme, we project the one-best predictions of English CRF to the foreign side through word alignments, then include the newly “tagged” foreign data as additional training data to a standard CRF in the foreign language. Rather than projecting labels on a per-word basis, Yarowsky and Ngai (2001) also explored an alternative method for noun-phrase (NP) bracketing task that amounts to projecting the spans of NPs based on the observation that individual NPs tend to retain their sequential spans across translations. We experimented with the same method for NER, but found that this method of projecting the NE spans does not help in reducing noise and actually lowers model performance.

Besides the difference in projecting expectations rather than hard labels, our method and the “project-then-train” scheme also differ by optimizing different objectives: CRF optimizes maximum conditional likelihood of the observed label sequence, whereas GE minimizes squared error between model’s expectation and “hard” expectation based on the observed label sequence. In the case where squared error loss is replaced with a KL-divergence loss, GE has the same effect as marginalizing out all positions with unknown projected labels, allowing more robust learning of uncertainties in the model. As we will show in the experimen-

	O	PER	LOC	ORG	GPE
O	291339	391	141	1281	221
PER	1263	6721	5	56	73
LOC	409	23	546	123	133
ORG	2423	143	52	8387	196
GPE	566	239	69	668	6604
	O	PER	LOC	ORG	MISC
O	81209	24	38	155	103
PER	77	5725	41	69	10
LOC	49	40	3743	127	60
ORG	178	102	142	4075	91
MISC	175	41	30	114	1826

Table 1: Raw counts in the error confusion matrix of English CRF models. Top table contains the counts on OntoNotes test data, and bottom table contains CoNLL-03 test data counts. Rows are the true labels and columns are the observed labels. For example, item at row 2, column 3 of the top table reads: we observed 5 times where the true label should be PERSON, but English CRF model output label LOCATION.

tal results in Section 4.2, soft projection in combination of the GE objective significantly outperforms the project-then-train style CRF training scheme.

3.3 Source-side noise

An additional source of noise comes from errors generated by the source-side English CRF models. We know that the English CRF models gives F_1 score of 81.68% on the OntoNotes dataset for English-Chinese experiment, and 90.45% on the CoNLL-03 dataset for English-German experiment. We present a simple way of modeling English-side noise by picturing the following process: the labels assigned by the English CRF model (denoted as \mathbf{y}) are some noised version of the true labels (denoted as \mathbf{y}^*). We can recover the probability of the true labels by marginalizing over the observed labels: $P(\mathbf{y}^*|\mathbf{x}) = \sum_{\mathbf{y}} P(\mathbf{y}^*|\mathbf{y}) * P(\mathbf{y}|\mathbf{x})$. $P(\mathbf{y}|\mathbf{x})$ is the posterior probabilities given by the CRF model, and we can approximate $P(\mathbf{y}^*|\mathbf{y})$ by the column-normalized error confusion matrix shown in Table 1. This source-side noise model is likely to be overly simplistic. Generally speaking, we could build much more sophisticated noising model for the source-side, possibly conditioning on context, or capturing higher-order label sequences.

4 Experiments

We conduct experiments on Chinese and German NER. We evaluate CLiPER in two learning settings: weakly supervised and semi-supervised. In the weakly supervised setting, we simulate the condition of having no labeled training data, and evaluate the model learned from bitext alone. We then vary the amount of labeled data available to the model, and examine the model’s learning curve. In the semi-supervised setting, we assume our model has access to the full labeled data; our goal is to improve performance of the supervised method by learning from additional bitext.

4.1 Dataset and setup

We used the latest version of Stanford NER Toolkit⁷ as our base CRF model in all experiments. Features for English, Chinese and German CRFs are documented extensively in (Che et al., 2013) and (Faruqui and Padó, 2010) and omitted here for brevity. It is worth noting that the current Stanford NER models include recent improvements from semi-supervised learning approaches that induces distributional similarity features from large word clusters. These models represent the current state-of-the-art in supervised methods, and serve as a very strong baseline.

For Chinese NER experiments, we follow the same setup as Che et al. (2013) to evaluate on the latest OntoNotes (v4.0) corpus (Hovy et al., 2006).⁸ A total of 8,249 sentences from the parallel Chinese and English Penn Treebank portion⁹ are reserved for evaluation. Odd-numbered documents are used as development set, and even-numbered documents are held out as blind test set. The rest of OntoNotes annotated with NER tags are used to train the English and Chinese CRF base taggers. There are about 16k and 39k labeled sentences for Chinese and English training, respectively. The English CRF tagger trained on this training corpus gives F₁ score of 81.68% on the OntoNotes test set. Four entity types¹⁰ are used for both Chinese and English with a IO tagging scheme.¹¹ The English-Chinese

bitext comes from the Foreign Broadcast Information Service corpus (FBIS).¹² We randomly sampled 80k parallel sentence pairs to use as bitext in our experiments. It is first sentence aligned using the Champollion Tool Kit,¹³ then word aligned with the BerkeleyAligner.¹⁴

For German NER experiments, we evaluate using the standard CoNLL-03 NER corpus (Sang and Meulder, 2003). The labeled training set has 12k and 15k sentences, containing four entity types.¹⁵ An English CRF model is also trained on the CoNLL-03 English data with the same entity types. For bitext, we used a randomly sampled set of 40k parallel sentences from the de-en portion of the *News Commentary* dataset.¹⁶ The English CRF tagger trained on CoNLL-03 English training corpus gives F₁ score of 90.4% on the CoNLL-03 test set.

We report typed entity precision (P), recall (R) and F₁ score. Statistical significance tests are done using a paired bootstrap resampling method with 1000 iterations, averaged over 5 runs. We compare against three recently approaches that were introduced in Section 2. They are: semi-supervised learning method using factored bilingual models with Gibbs sampling (Wang et al., 2013a); bilingual NER using Integer Linear Programming (ILP) with bilingual constraints, by (Che et al., 2013); and constraint-driven bilingual-reranking approach (Burkett et al., 2010). The code from (Che et al., 2013) and (Wang et al., 2013a) are publicly available.¹⁷ Code from (Burkett et al., 2010) is obtained through personal communications.

Since the objective function in Eqn. 2 is non-convex, we adopted the early stopping training scheme from (Turian et al., 2010) as the following: after each iteration in L-BFGS training, the model

(Ramshaw and Marcus, 1999), because when projected across swapping word alignments, the “B-” and “I-” tag distinction may not be well-preserved and may introduce additional noise.

¹²The FBIS corpus is a collection of radio news casts and contains translations of openly available news and information from media sources outside the United States. The LDC catalogue No. is LDC2003E14.

¹³champollion.sourceforge.net

¹⁴code.google.com/p/berkeleyaligner

¹⁵PERSON, LOCATION, ORGANIZATION and MISCELLANEOUS.

¹⁶<http://www.statmt.org/wmt13/training-parallel-nc-v8.tgz>

¹⁷<https://github.com/stanfordnlp/CoreNLP>

⁷<http://www-nlp.stanford.edu/ner>

⁸LDC catalogue No.: LDC2011T03

⁹File numbers: chtb.0001-0325, ectb.1001-1078

¹⁰PERSON, LOCATION, ORGANIZATION and GPE.

¹¹We did not adopt the commonly seen BIO tagging scheme

is evaluated against the development set; the training procedure is terminated if no improvements have been made in 20 iterations.

4.2 Weakly supervised results

Figure 2a and 2b show results of weakly supervised learning experiments. Quite remarkably, on Chinese test set, our proposed method (CLiPER) achieves a F_1 score of 64.4% with 80k bitext, when no labeled training data is used. In contrast, the supervised CRF baseline would require as much as 12k labeled sentences to attain the same accuracy. Results on the German test set is less striking. With no labeled data and 40k of bitext, CLiPER performs at F_1 of 60.0%, the equivalent of using 1.5k labeled examples in the supervised setting. When combined with 1k labeled examples, performance of CLiPER reaches 69%, a gain of over 5% absolute over supervised CRF. We also notice that supervised CRF model learns much faster in German than Chinese. This result is not too surprising, since it is well recognized that Chinese NER is more challenging than German or English. The best supervised results for Chinese is 10-20% (F_1 score) behind best German and English supervised results. Chinese NER relies more on lexicalized features, and therefore needs more labeled data to achieve good coverage. The results suggest that CLiPER seems to be very effective at transferring lexical knowledge from English to Chinese.

Figure 2c and 2d compares soft GE projection with hard GE projection and the “project-then-train” style CRF training scheme (*cf.* Section 3.2). We observe that both soft and hard GE projection significantly outperform the “project-then-train” style training scheme. The difference is especially pronounced on the Chinese results when fewer labeled examples are available. Soft projection gives better accuracy than hard projection when no labeled data is available, and also has a faster learning rate.

Incorporating source-side noise using the method described in Section 3.3 gives a small improvement on Chinese with supervised data, increasing F_1 score from 64.40% to 65.50%. This improvement is statistically significant at 92% confidence interval. However, on the German data, we observe a tiny decrease with no statistical significance in F_1 score, dropping from 59.88% to 59.66%. A likely explanation of the difference is that the English CRF

model in the English-Chinese experiment, which is trained on OntoNotes data, has a much higher error rate (18.32%) than the English CRF model in the English-German experiment trained on CoNLL-03 (9.55%). Therefore, modeling noise in the English-Chinese case is likely to have a greater effect than the English-German case.

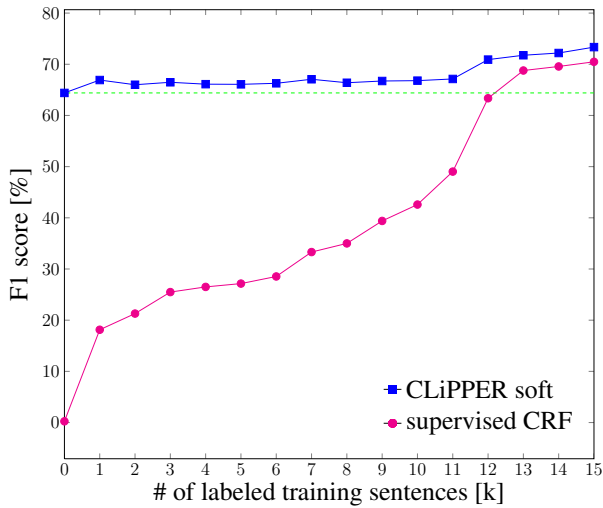
4.3 Semi-supervised results

In the semi-supervised experiments, we let the CRF model use the full set of labeled examples in addition to the unlabeled bitext. Results on the test set are shown in Table 2. All semi-supervised baselines are tested with the same number of unlabeled bitext as CLiPER in each language. The “project-then-train” semi-supervised training scheme severely hurts performance on Chinese, but gives a small improvement on German. Moreover, on Chinese it learns to achieve high precision but at a significant loss in recall. On German its behavior is the opposite. Such drastic and erratic imbalance suggest that this method is not robust or reliable. The other three semi-supervised baselines (row 3-5) all show improvements over the CRF baseline, consistent with their reported results. CLiPER_s gives the best results on both Chinese and German, yielding statistically significant improvements over all baselines except for CWD13 on German. The hard projection version of CLiPER also gives sizable gain over CRF. However, in comparison, CLiPER_s is superior.

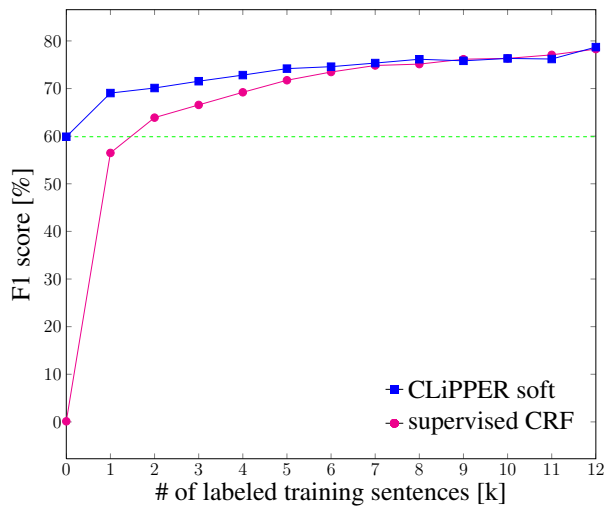
The improvements of CLiPER_s over CRF on Chinese test set is over 2.8% in absolute F_1 . The improvement over CRF on German is almost a percent. To our knowledge, these are the best reported numbers on the OntoNotes Chinese and CoNLL-03 German datasets.

4.4 Efficiency

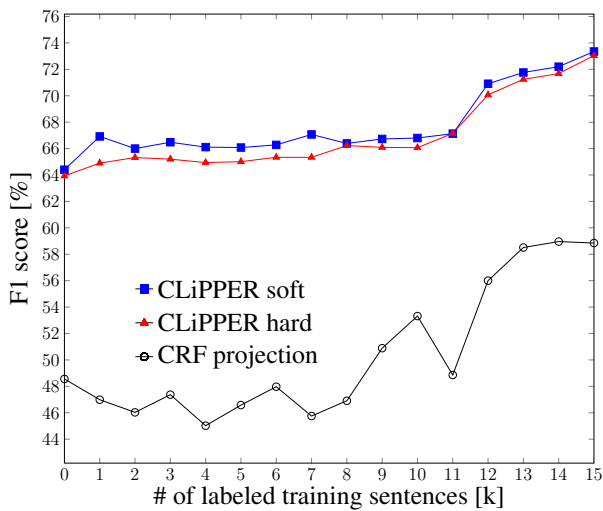
Another advantage of our proposed approach is efficiency. Because we eliminated the previous multi-stage “uptraining” paradigm, but instead integrating the semi-supervised and supervised objective into one joint objective, we are able to attain significant speed improvements over all methods except CRF_{ptt}. Table 3 shows the required training time.



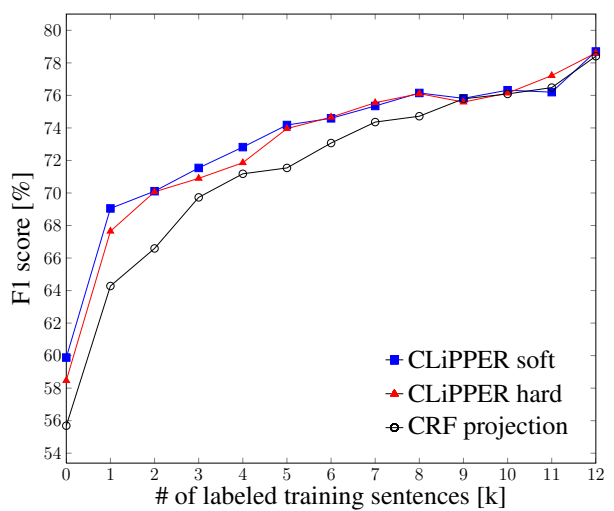
(a) Chinese Test



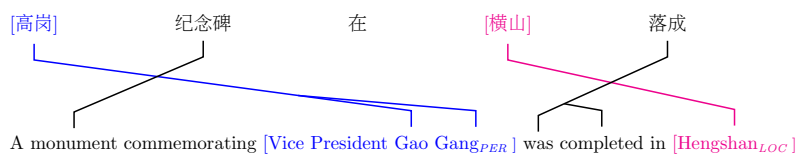
(b) German Test



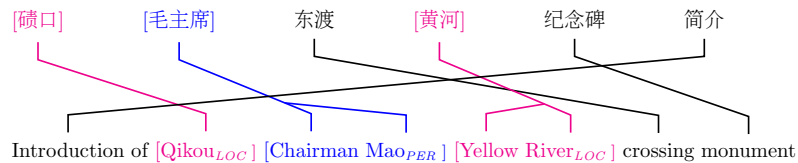
(c) Soft vs. Hard on Chinese Test



(d) Soft vs. Hard on German Test



(e) Word preceding “monument” is PERSON



(f) Word preceding “monument” is LOCATION

Figure 2: Top four figures show performance curves of CLiPPER with varying amounts of available labeled training data in a weakly supervised setting. Vertical axes show the F₁ score on the test set. Performance curves of supervised CRF and “project-then-train” CRF are plotted for comparison. Bottom two figures are examples of aligned sentence pairs in Chinese and English.

	Chinese			German		
	P	R	F ₁	P	R	F ₁
CRF	79.09	63.59	70.50	86.69	71.30	78.25
CRF _{ptt}	84.01	45.29	58.85	81.50	75.56	78.41
BPBK10	79.25	65.67	71.83	84.00	72.17	77.64
CWD13	81.31	65.50	72.55	85.99	72.98	78.95
WCD13a	80.31	65.78	72.33	85.98	72.37	78.59
WCD13b	78.55	66.54	72.05	85.19	72.98	78.62
CLiPER _h	83.67	64.80	73.04 ^{‡†}	86.52	72.02	78.61 [*]
CLiPER _s	82.57	65.99	73.35 ^{‡†*}	87.11	72.56	79.17 ^{‡†*} _{*§}

Table 2: Test set Chinese, German NER results. Best number of each column is highlighted in bold. CRF is the supervised baseline. CRF_{ptt} is the “project-then-train” semi-supervised scheme for CRF. BPBK10 is (Burkett et al., 2010), WCD13 is (Wang et al., 2013a), CWD13A is (Che et al., 2013), and WCD13B is (Wang et al., 2013b). CLiPER_s and CLiPER_h are the soft and hard projections. § indicates F₁ scores that are statistically significantly better than CRF baseline at 99.5% confidence level; * marks significance over CRF_{ptt} with 99.5% confidence; † and ‡ marks significance over WCD13 with 99.9% and 94% confidence; and ◊ marks significance over CWD13 with 99.7% confidence; * marks significance over BPBK10 with 99.9% confidence.

5 Discussions

Figure 2e and 2f give two examples of cross-lingual projection methods in action. Both examples have a named entity that immediately proceeds the word “纪念碑” (monument) in the Chinese sentence. In Figure 2e, the word “高岗” has literal meaning of *a hillock located at a high position*, which also happens to be the name of a former vice president of China. Without having previously observed this word as a person name in the labeled training data, the CRF model does not have enough evidence to believe that this is a PERSON, instead of LOCATION. But the aligned words in English (“Gao Gang”) are clearly part of a person name as they were preceded by a title (“Vice President”). The English model has high expectation that the aligned Chinese word of “Gao Gang” is also a PERSON. Therefore, projecting the English expectations to Chinese provides a strong clue to help disambiguating this word. Figure 2f gives another example: the word “黄河”(Huang He, the Yellow River of China) can

	Chinese	German
CRF	19m30s	7m15s
CRF _{ptt}	34m2s	12m45s
WCD13	3h17m	1h1m
CWD13a	16h42m	4h49m
CWD13b	16h42m	4h49m
BPBK10	6h16m	2h42m
CLiPER _h	1h28m	16m30s
CLiPER _s	1h40m	18m51s

Table 3: Timing stats during model training.

be confused with a person name since “黄”(Huang or Hwang) is also a common Chinese last name.¹⁸. Again, knowing the translation in English, which has the indicative word “River” in it, helps disambiguation.

The CRF_{ptt} and CLiPER_h methods successfully labeled these two examples correctly, but failed to produce the correct label for the example in Figure 1. On the other hand, a model trained with the CLiPER_s method does correctly label both entities in Figure 1, demonstrating the merits of the soft projection method.

6 Conclusion

We introduced a domain and language independent semi-supervised method for training discriminative models by projecting expectations across bitext. Experiments on Chinese and German NER show that our method, learned over bitext alone, can rival performance of supervised models trained with thousands of labeled examples. Furthermore, applying our method in a setting where all labeled examples are available also shows improvements over state-of-the-art supervised methods. Our experiments also showed that soft expectation projection is more favorable to hard projection. This technique can be generalized to all sequence labeling tasks, and can be extended to include more complex constraints. For future work, we plan to apply this method to more language pairs and also explore data selection strategies and modeling alignment uncertainties.

¹⁸In fact, a people search of the name 黄河 on the most popular Chinese social network (renren.com) returns over 13,000 matches.

Acknowledgments

The authors would like to thank Jennifer Gillenwater for a discussion that inspired this work, Behrang Mohit and Nathan Schneider for their help with the Arabic NER data, and David Burkett for providing the source code of their work for comparison. We would also like to thank editor Lillian Lee and the three anonymous reviewers for their valuable comments and suggestions. We gratefully acknowledge the support of the U.S. Defense Advanced Research Projects Agency (DARPA) Broad Operational Language Translation (BOLT) program through IBM. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of DARPA, or the US government.

References

- Hiyan Alshawi, Srinivas Bangalore, and Shona Douglas. 2000. Head-transducer models for speech translation and their automatic acquisition from bilingual data. *Machine Translation*, 15.
- Rie Kubota Ando and Tong Zhang. 2005. A high-performance semi-supervised learning method for text chunking. In *Proceedings of ACL*.
- Kedar Bellare, Gregory Druck, and Andrew McCallum. 2009. Alternating projections for learning with expectation constraints. In *Proceedings of UAI*.
- Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of COLT*.
- David Burkett and Dan Klein. 2008. Two languages are better than one (for syntactic parsing). In *Proceedings of EMNLP*.
- David Burkett, Slav Petrov, John Blitzer, and Dan Klein. 2010. Learning better monolingual models with unannotated bilingual text. In *Proceedings of CoNLL*.
- Andrew Carlson, Justin Betteridge, Richard C. Wang, Esdevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of WSDM*.
- Ming-Wei Chang, Lev Ratinov, and Dan Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *Proceedings of ACL*.
- Wanxiang Che, Mengqiu Wang, and Christopher D. Manning. 2013. Named entity recognition with bilingual constraints. In *Proceedings of NAACL*.
- Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of EMNLP*.
- Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of ACL*.
- Gregory Druck and Andrew McCallum. 2010. High-performance semi-supervised learning using discriminatively constrained generative models. In *Proceedings of ICML*.
- Gregory Druck, Gideon Mann, and Andrew McCallum. 2007. Leveraging existing resources using generalized expectation criteria. In *Proceedings of NIPS Workshop on Learning Problem Design*.
- Gregory Druck, Burr Settles, and Andrew McCallum. 2009. Active learning by labeling features. In *Proceedings of EMNLP*.
- Gregory Druck. 2011. *Generalized Expectation Criteria for Lightly Supervised Learning*. Ph.D. thesis, University of Massachusetts Amherst.
- Manaal Faruqui and Sebastian Padó. 2010. Training and evaluating a German named entity recognizer with semantic generalization. In *Proceedings of KONVENS*.
- Jenny Rose Finkel, Christopher D. Manning, and Andrew Y. Ng. 2006. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In *Proceedings of EMNLP*.
- Victoria Fossum and Steven Abney. 2005. Automatically inducing a part-of-speech tagger by projecting from multiple source languages across aligned corpora. In *Proceedings of IJCNLP*.
- Kuzman Ganchev and Dipanjan Das. 2013. Cross-lingual discriminative learning of sequence models with posterior regularization. In *Proceedings of EMNLP*.
- Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of ACL*.
- Kuzman Ganchev, Jo ao Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *JMLR*, 10:2001–2049.
- Andrew B. Goldberg. 2010. *New Directions in Semi-supervised Learning*. Ph.D. thesis, University of Wisconsin-Madison.
- Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: the 90% solution. In *Proceedings of NAACL-HLT*.
- Dan Klein. 2005. *The Unsupervised Learning of Natural Language Structure*. Ph.D. thesis, Stanford University.
- John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

- Zhifei Li and Jason Eisner. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proceedings of EMNLP*.
- Shen Li, Jo ao Graça, and Ben Taskar. 2012. Wiki-ly supervised part-of-speech tagging. In *Proceedings of EMNLP-CoNLL*.
- Percy Liang. 2005. Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology.
- Gideon Mann and Andrew McCallum. 2010. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *JMLR*, 11:955–984.
- David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of NAACL-HLT*.
- Tahira Naseem, Benjamin Snyder, Jacob Eisenstein, and Regina Barzilay. 2009. Multilingual part-of-speech tagging: Two unsupervised approaches. *JAIR*, 36:1076–9757.
- Slav Petrov, Pi-Chuan Chang, Michael Ringgaard, and Hiyan Alshawi. 2010. Uptraining for accurate deterministic question parsing. In *Proceedings of EMNLP*.
- Lance A. Ramshaw and Mitchell P. Marcus. 1999. Text chunking using transformation-based learning. *Natural Language Processing Using Very Large Corpora*, 11:157–176.
- Rajhans Samdani, Ming-Wei Chang, and Dan Roth. 2012. Unified expectation maximization. In *Proceedings of NAACL*.
- 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 CoNLL*.
- Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. 2005. A co-regularization approach to semi-supervised learning with multiple views. In *Proceedings of ICML Workshop on Learning with Multiple Views, International Conference on Machine Learning*.
- Noah A. Smith. 2006. *Novel Estimation Methods for Unsupervised Discovery of Latent Structure in Natural Language Text*. Ph.D. thesis, Johns Hopkins University.
- Benjamin Snyder, Tahira Naseem, and Regina Barzilay. 2009. Unsupervised multilingual grammar induction. In *Proceedings of ACL*.
- Jun Suzuki and Hideki Isozaki. 2008. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In *Proceedings of ACL*.
- Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. In *Proceedings of ACL*.
- Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of ACL*.
- Mengqiu Wang, Wanxiang Che, and Christopher D. Manning. 2013a. Effective bilingual constraints for semi-supervised learning of named entity recognizers. In *Proceedings of AAAI*.
- Mengqiu Wang, Wanxiang Che, and Christopher D. Manning. 2013b. Joint word alignment and bilingual named entity recognition using dual decomposition. In *Proceedings of ACL*.
- Chenhai Xi and Rebecca Hwa. 2005. A backoff model for bootstrapping resources for non-english languages. In *Proceedings of HLT-EMNLP*.
- David Yarowsky and Grace Ngai. 2001. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proceedings of NAACL*.
- David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of ACL*.

Back to Basics for Monolingual Alignment: Exploiting Word Similarity and Contextual Evidence

Md Arafat Sultan[†], Steven Bethard[‡] and Tamara Sumner[†]

[†]Institute of Cognitive Science and Department of Computer Science
University of Colorado Boulder

[‡]Department of Computer and Information Sciences
University of Alabama at Birmingham

arafat.sultan@colorado.edu, bethard@cis.uab.edu, sumner@colorado.edu

Abstract

We present a simple, easy-to-replicate monolingual aligner that demonstrates state-of-the-art performance while relying on almost no supervision and a very small number of external resources. Based on the hypothesis that words with similar meanings represent potential pairs for alignment if located in similar contexts, we propose a system that operates by finding such pairs. In two intrinsic evaluations on alignment test data, our system achieves F_1 scores of 88–92%, demonstrating 1–3% absolute improvement over the previous best system. Moreover, in two extrinsic evaluations our aligner outperforms existing aligners, and even a naive application of the aligner approaches state-of-the-art performance in each extrinsic task.

1 Introduction

Monolingual alignment is the task of discovering and aligning similar semantic units in a pair of sentences expressed in a natural language. Such alignments provide valuable information regarding how and to what extent the two sentences are related. Consequently, alignment is a central component of a number of important tasks involving text comparison: textual entailment recognition, textual similarity identification, paraphrase detection, question answering and text summarization, to name a few.

The high utility of monolingual alignment has spawned significant research on the topic in the recent past. Major efforts that have treated alignment as a standalone problem (MacCartney et al., 2008; Thadani and McKeown, 2011; Yao et al., 2013a) are

primarily supervised, thanks to the manually aligned corpus with training and test sets from Microsoft Research (Brockett, 2007). Primary concerns of such work include both quality and speed, due to the fact that alignment is frequently a component of larger NLP tasks.

Driven by similar motivations, we seek to devise a lightweight, easy-to-construct aligner that produces high-quality output and is applicable to various end tasks. Amid a variety of problem formulations and ingenious approaches to alignment, we take a step back and examine closely the effectiveness of two frequently made assumptions: 1) Related semantic units in two sentences must be similar or related in their meaning, and 2) Commonalities in their semantic contexts in the respective sentences provide additional evidence of their relatedness (MacCartney et al., 2008; Thadani and McKeown, 2011; Yao et al., 2013a; Yao et al., 2013b). Alignment, based solely on these two assumptions, reduces to finding the best combination of pairs of similar semantic units in similar contexts.

Exploiting existing resources to identify similarity of semantic units, we search for robust techniques to identify contextual commonalities. Dependency trees are a commonly used structure for this purpose. While they remain a central part of our aligner, we expand the horizons of dependency-based alignment beyond exact matching by systematically exploiting the notion of “type equivalence” with a small hand-crafted set of equivalent dependency types. In addition, we augment dependency-based alignment with surface-level text analysis.

While phrasal alignments are important and have

been investigated in multiple studies, we focus primarily on word alignments (which have been shown to form the vast majority of alignments ($\geq 95\%$) in multiple human-annotated corpora (Yao et al., 2013b)), keeping the framework flexible enough to allow incorporation of phrasal alignments in future.

Evaluation of our aligner on the benchmark dataset reported in (Brockett, 2007) shows an F_1 score of 91.7%: a 3.1% absolute improvement over the previous best system (Yao et al., 2013a), corresponding to a 27.2% error reduction. It shows superior performance also on the dataset reported in (Thadani et al., 2012). Additionally, we present results of two extrinsic evaluations, namely textual similarity identification and paraphrase detection. Our aligner not only outperforms existing aligners in each task, but also approaches top systems for the extrinsic tasks.

2 Background

Monolingual alignment has been applied to various NLP tasks including textual entailment recognition (Hickl et al., 2006; Hickl and Bensley, 2007), paraphrase identification (Das and Smith, 2009; Madhani et al., 2012), and textual similarity assessment (Bär et al., 2012; Han et al., 2013) – in some cases explicitly, i.e., as a separate module. But many such systems resort to simplistic and/or ad-hoc strategies for alignment and in most such work, the alignment modules were not separately evaluated on alignment benchmarks, making their direct assessment difficult.

With the introduction of the MSR alignment corpus (Brockett, 2007) developed from the second Recognizing Textual Entailment challenge data (Bar-Haim et al., 2006), direct evaluation and comparison of aligners became possible. The first aligner trained and evaluated on the corpus was a phrasal aligner called MANLI (MacCartney et al., 2008). It represents alignments as sets of different *edit* operations (where a sequence of edits turns one input sentence into the other) and finds an optimal set of edits via a simulated annealing search. Weights of different edit features are learned from the training set of the MSR alignment corpus using a perceptron learning algorithm. MANLI incorporates only shallow features characterizing contextual similarity: relative positions of the two phrases being aligned (or not) in the two sentences and boolean features representing

whether or not the preceding and following tokens of the two phrases are similar.

Thadani and McKeown (2011) substituted MANLI’s simulated annealing-based decoding with integer linear programming, and achieved a considerable speed-up. More importantly for our discussion, they found contextual evidence in the form of syntactic constraints useful in better aligning stop words. Thadani et al. (2012) further extended the model by adding features characterizing dependency *arc edits*, effectively bringing stronger influence of contextual similarity into alignment decisions. Again the performance improved consequently.

The most successful aligner to date both in terms of accuracy and speed, called JacanaAlign, was developed by Yao et al. (2013a). In contrast to the earlier systems, JacanaAlign is a word aligner that formulates alignment as a sequence labeling problem. Each word in the source sentence is labeled with the corresponding target word index if an alignment is found. It employs a conditional random field to assign the labels and uses a feature set similar to MANLI’s in terms of the information they encode (with some extensions). Contextual features include only semantic match of the left and the right neighbors of the two words and their POS tags. Even though JacanaAlign outperformed the MANLI enhancements despite having less contextual features, it is difficult to compare the role of context in the two models because of the large paradigmatic disparity. An extension of JacanaAlign was proposed for phrasal alignments in (Yao et al., 2013b), but the contextual features remained largely unchanged.

Noticeable in all the above systems is the use of contextual evidence as a feature for alignment, but in our opinion, not to an extent sufficient to harness its full potential. Even though deeper dependency-based modeling of contextual commonalities can be found in some other studies (Kouylekov and Magnini, 2005; Chambers et al., 2007; Chang et al., 2010; Yao et al., 2013c), we believe there is further scope for systematic exploitation of contextual evidence for alignment, which we aim to do in this work.

On the contrary, word semantic similarity has been a central component of most aligners; various measures of word similarity have been utilized, including string similarity, resource-based similarity (derived from one or more lexical resources like WordNet)

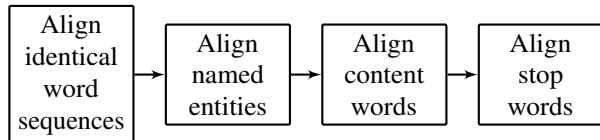


Figure 1: System overview

and distributional similarity (computed from word co-occurrence statistics in large corpora). An important trade-off between precision, coverage and speed exists here and aligners commonly rely on only a subset of these measures (Thadani and McKeown, 2011; Yao et al., 2013a). We use the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013), which is a large resource of lexical and phrasal paraphrases constructed using bilingual pivoting (Bannard and Callison-Burch, 2005) over large parallel corpora.

3 System

Our system operates as a pipeline of alignment modules that differ in the types of word pairs they align. Figure 1 is a simplistic representation of the pipeline. Each module makes use of contextual evidence to make alignment decisions. In addition, the last two modules are informed by a word semantic similarity algorithm. Because of their phrasal nature, we treat named entities separately from other content words. The rationale behind the order in which the modules are arranged is discussed later in this section (3.3.5).

Before discussing each alignment module in detail, we describe the system components that identify word and contextual similarity.

3.1 Word Similarity

The ability to correctly identify semantic similarity between words is crucial to our aligner, since contextual evidence is important only for similar words. Instead of treating word similarity as a continuous variable, we define three levels of similarity.

The first level is an exact word or lemma match which is represented by a similarity score of 1. The second level represents semantic similarity between two terms which are not identical. To identify such word pairs, we employ the Paraphrase Database (PPDB)¹. We use the largest (XXXL) of the PPDB’s lexical paraphrase packages and treat all pairs identically by ignoring the accompanying statistics. We

¹<http://paraphrase.org>

customize the resource by removing pairs of identical words or lemmas and adding lemmatized forms of the remaining pairs. For now, we use the term *ppdbSim* to refer to the similarity of each word pair in this modified version of PPDB (which is a value in $(0, 1)$) and later explain how we determine it (Section 3.3.5). Finally, any pair of different words which is absent in PPDB is assigned a zero similarity score.

3.2 Extracting Contextual Evidence

Our alignment modules collect contextual evidence from two complementary sources: syntactic dependencies and words occurring within a small textual vicinity of the two words to be aligned. The application of each kind assumes a common principle of *minimal* evidence. Formally, given two input sentences S and T , we consider two words $s \in S$ and $t \in T$ to form a candidate pair for alignment if $\exists r_s \in S$ and $\exists r_t \in T$ such that:

1. $(s, t) \in \mathcal{R}_{Sim}$ and $(r_s, r_t) \in \mathcal{R}_{Sim}$, where \mathcal{R}_{Sim} is a binary relation indicating *sufficient* semantic relatedness between the members of each pair ($\geq ppdbSim$ in our case).
2. $(s, r_s) \in \mathcal{R}_{C_1}$ and $(t, r_t) \in \mathcal{R}_{C_2}$, such that $\mathcal{R}_{C_1} \approx \mathcal{R}_{C_2}$; where \mathcal{R}_{C_1} and \mathcal{R}_{C_2} are binary relations representing specific types of contextual relationships between two words in a sentence (e.g., an *nsubj* dependency between a verb and a noun). The symbol \approx represents equivalence between two relationships, including identity.

Note that the minimal-evidence assumption holds a single piece of contextual evidence as sufficient support for a potential alignment; but as we discuss later in this section, an evidence for word pair (s, t) (where $s \in S$ and $t \in S$) may not lead to an alignment if there exists a competing pair (s', t) or (s, t') with more evidence (where $s' \in S$ and $t' \in T$).

In the rest of this section, we elaborate the different forms of contextual relationships we exploit along with the notion of equivalence between relationships.

3.2.1 Syntactic Dependencies

Dependencies can be important sources of contextual evidence. Two *nsubj* children r_s and r_t of two verbs $s \in S$ and $t \in T$, for example, provide evidence for not only an (s, t) alignment, but

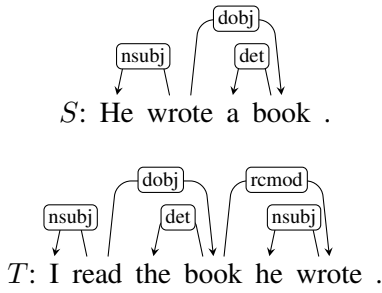


Figure 2: Equivalent dependency types: *dobj* and *rcmod*

also an (r_s, r_t) alignment if $(s, t) \in \mathfrak{R}_{Sim}$ and $(r_s, r_t) \in \mathfrak{R}_{Sim}$. (We adopt the Stanford typed dependencies (de Marneffe and Manning, 2008).)

Moreover, dependency types can exhibit equivalence; consider the two sentences in Figure 2. The *dobj* dependency in S is equivalent to the *rcmod* dependency in T ($dobj \approx rcmod$, following our earlier notation) since they represent the same semantic relation between identical word pairs in the two sentences. To be able to use such evidence for alignment, we need to go beyond exact matching of dependencies and develop a mapping among dependency types that encodes such equivalence. Note also that the parent-child roles are opposite for the two dependency types in the above example, a scenario that such a mapping must accommodate.

The four possible such scenarios regarding parent-child orientations are shown in Figure 3. If $(s, t) \in \mathfrak{R}_{Sim}$ and $(r_s, r_t) \in \mathfrak{R}_{Sim}$ (represented by bidirectional arrows), then each orientation represents a set of possible ways in which the S and T dependencies (unidirectional arrows) can provide evidence of similarity between the contexts of s in S and t in T . Each such set comprises equivalent dependency type pairs for that orientation. In the example of Figure 2, $(dobj, rcmod)$ is such a pair for orientation (c), given $s = t = \text{“wrote”}$ and $r_s = r_t = \text{“book”}$.

We apply the notion of dependency type equivalence to intra-category alignment of content words in four major lexical categories: *verbs*, *nouns*, *adjectives* and *adverbs* (the Stanford POS tagger (Toutanova et al., 2003) is used to identify the categories). Table 1 shows dependency type equivalences for each lexical category of s and t .

The ‘ \leftarrow ’ sign on column 5 of some rows represents a duplication of the column 4 content of the

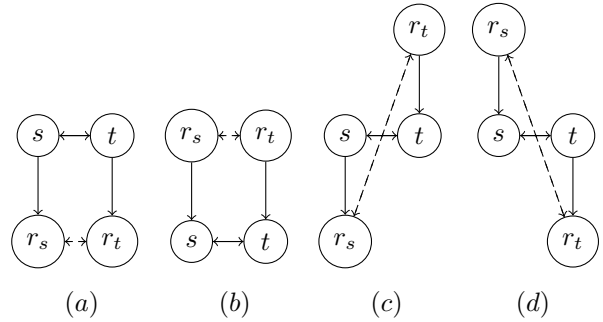


Figure 3: Parent-child orientations in dependencies

same row. For each row, columns 4 and 5 show two sets of dependency types; each member of the first is equivalent to each member of the second for the current orientation (column 1) and lexical categories of the associated words (columns 2 and 3). For example, row 2 represents the fact that an *agent* relation (between s and r_s ; s is the parent) is equivalent to an *nsubj* relation (between t and r_t ; t is the parent).

Note that the equivalences are fundamentally redundant across different orientations. For example, row 2 (which is presented as an instance of orientation (a)) can also be presented as an instance of orientation (b) with $POS(s)=POS(t)=noun$ and $POS(r_s)=POS(r_t)=verb$. We avoid such redundancy for compactness. For the same reason, the equivalence of *dobj* and *rcmod* in Figure 2 is shown in the table only as an instance of orientation (c) and not as an instance of orientation (d) (in general, this is why orientations (b) and (d) are absent in the table).

We present dependency-based contextual evidence extraction in Algorithm 1. (The Stanford dependency parser (de Marneffe et al., 2006) is used to extract the dependencies.) Given a word pair (s_i, t_j) from the input sentences S and T , it collects contextual evidence (as indexes of r_{s_i} and r_{t_j} with a positive similarity) for each matching row in Table 1. An exact match of the two dependencies is also considered a piece of evidence. Note that Table 1 only considers content word pairs (s_i, t_j) such that $POS(s_i)=POS(t_j)$, but as 90% of all content word alignments in the MSR alignment *dev* set are within the same lexical category, this is a reasonable set to start with.

3.2.2 Textual Neighborhood

While equivalent dependencies can provide strong contextual evidence, they can not ensure high recall because, a) the ability to accurately extract depen-

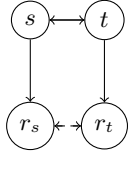
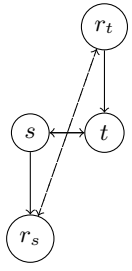
Orientation	POS(s, t)	POS(r_s, r_t)	S Dependency Types	T Dependency Types		
 <p>(a)</p>	verb	verb	{purpcl, xcomp}	←		
		noun	noun	{agent, nsubj, xsubj} {dobj, nsubjpass, rel} {tmod, prep_in, prep_at, prep_on} {iobj, prep_to}	←	
			noun	{pos, nn, prep_of, prep_in, prep_at, prep_for}	←	
	 <p>(c)</p>	verb	verb	{infmod, partmod, rcmod}	←	
			noun	noun	{pos, nn, prep_of, prep_in, prep_at, prep_for}	←
			adjective	adjective	{amod, rcmod}	←
noun		verb	verb	{conj_and} {conj_or} {conj_nor}	←	
		noun	noun	{dobj, nsubjpass, rel}	{infmod, partmod, rcmod}	
		adjective	adjective	{acomp}	{cop, csubj}	
		noun	noun	{conj_and} {conj_or} {conj_nor}	←	
		adjective	adjective	{amod, rcmod}	{nsubj}	
		adverb	adverb	{conj_and} {conj_or} {conj_nor}	←	

Table 1: Equivalent dependency structures

Algorithm 1: $depContext(S, T, i, j, EQ)$

Input:

1. S, T : Sentences to be aligned
2. i : Index of a word in S
3. j : Index of a word in T
4. EQ : Dependency type equivalences (Table 1)

Output: $context = \{(k, l)\}$: pairs of word indexes

- 1 $context \leftarrow \{(k, l) : wordSim(s_k, t_l) > 0$
- 2 $\wedge (i, k, \tau_s) \in dependencies(S)$
- 3 $\wedge (j, l, \tau_t) \in dependencies(T)$
- 4 $\wedge POS(s_i) = POS(t_j) \wedge POS(s_k) = POS(t_l)$
- 5 $\wedge (\tau_s = \tau_t$
- 6 $\quad \vee (POS(s_i), POS(s_k), \tau_s, \tau_t) \in EQ))\}$

dependencies is limited by the accuracy of the parser, and b) we investigate equivalence types for only inter-lexical-category alignment in this study. Therefore we apply an additional model of word context: the textual neighborhood of s in S and t in T .

Extraction of contextual evidence for content words from textual neighborhood is described in Algorithm 2. Like the dependency-based module, it accumulates evidence for each (s_i, t_j) pair by inspecting multiple pairs of neighboring words. But instead of aligning only words within a lexical category,

Algorithm 2: $textContext(S, T, i, j, STOP)$

Input:

1. S, T : Sentences to be aligned
2. i : Index of a word in S
3. j : Index of a word in T
4. $STOP$: A set of stop words

Output: $context = \{(k, l)\}$: pairs of word indexes

- 1 $C_i \leftarrow \{k : k \in [i - 3, i + 3] \wedge k \neq i \wedge s_k \notin STOP\}$
- 2 $C_j \leftarrow \{l : l \in [j - 3, j + 3] \wedge l \neq j \wedge t_l \notin STOP\}$
- 3 $context \leftarrow C_i \times C_j$

this module also performs inter-category alignment, considering content words within a (3, 3) window of s_i and t_j as neighbors. We implement relational equivalence (\approx) here by holding any two positions within the window equally contributive and mutually comparable as sources of contextual evidence.

3.3 The Alignment Algorithm

We now describe each alignment module in the pipeline and their sequence of operation.

3.3.1 Identical Word Sequences

The presence of a common word sequence in S and T is indicative of an (a) identical, and (b) con-

textually similar word in the other sentence for each word in the sequence. We observe the results of aligning identical words in such sequences of length n containing at least one content word. This simple heuristic demonstrates a high precision ($\approx 97\%$) on the MSR alignment *dev* set for $n \geq 2$, and therefore we consider membership in such sequences as the simplest form of contextual evidence in our system and align all identical word sequence pairs in S and T containing at least one content word. From this point on, we will refer to this module as *wsAlign*.

3.3.2 Named Entities

We align named entities separately to enable the alignment of full and partial mentions (and acronyms) of the same entity. We use the Stanford Named Entity Recognizer (Finkel et al., 2005) to identify named entities in S and T . After aligning the exact term matches, any unmatched term of a partial mention is aligned to all terms in the full mention. The module recognizes only first-letter acronyms and aligns an acronym to all terms in the full mention of the corresponding name.

Since named entities are instances of nouns, named entity alignment is also informed by contextual evidence (which we discuss in the next section), but happens before alignment of other generic content words. Parents (or children) of a named entity are simply the parents (or children) of its head word. We will refer to this module as a method named *neAlign* from this point on.

3.3.3 Content Words

Extraction of contextual evidence for promising content word pairs has already been discussed in Section 3.2, covering both dependency-based context and textual context.

Algorithm 3 (*cwDepAlign*) describes the dependency-based alignment process. For each potentially alignable pair (s_i, t_j) , the dependency-based context is extracted as described in Algorithm 1, and context similarity is calculated as the sum of the word similarities of the (s_k, t_l) context word pairs (lines 2-7). (The *wordSim* method returns a similarity score in $\{0, ppdbSim, 1\}$.) The alignment score of the (s_i, t_j) pair is then a weighted sum of word and contextual similarity (lines 8-12). (We discuss how the weights are set in Section

Algorithm 3: *cwDepAlign*($S, T, EQ, A_E, w, STOP$)

Input:

1. S, T : Sentences to be aligned
2. EQ : Dependency type equivalences (Table 1)
3. A_E : Already aligned word pair indexes
4. w : Weight of word similarity relative to contextual similarity
5. $STOP$: A set of stop words

Output: $A = \{(i, j)\}$: word index pairs of aligned words $\{(s_i, t_j)\}$ where $s_i \in S$ and $t_j \in T$

```

1  $\Psi \leftarrow \emptyset; \Lambda_\Psi \leftarrow \emptyset; \Phi \leftarrow \emptyset$ 
2 for  $s_i \in S, t_j \in T$  do
3   if  $s_i \notin STOP \wedge \neg \exists t_l : (i, l) \in A_E$ 
4      $\wedge t_j \notin STOP \wedge \neg \exists s_k : (k, j) \in A_E$ 
5      $\wedge wordSim(s_i, t_j) > 0$  then
6        $context \leftarrow depContext(S, T, i, j, EQ)$ 
7        $contextSim \leftarrow \sum_{(k,l) \in context} wordSim(s_k, t_l)$ 
8       if  $contextSim > 0$  then
9          $\Psi \leftarrow \Psi \cup \{(i, j)\}$ 
10         $\Lambda_\Psi(i, j) \leftarrow context$ 
11         $\Phi(i, j) \leftarrow w * wordSim(s_i, t_j)$ 
12         $+ (1 - w) * contextSim$ 
13 Linearize and sort  $\Psi$  in decreasing order of  $\Phi(i, j)$ 
14  $A \leftarrow \emptyset$ 
15 for  $(i, j) \in \Psi$  do
16   if  $\neg \exists l : (i, l) \in A$ 
17      $\wedge \neg \exists k : (k, j) \in A$  then
18      $A \leftarrow A \cup \{(i, j)\}$ 
19   for  $(k, l) \in \Lambda_\Psi(i, j)$  do
20     if  $\neg \exists q : (k, q) \in A \cup A_E$ 
21      $\wedge \neg \exists p : (p, l) \in A \cup A_E$  then
22      $A \leftarrow A \cup \{(k, l)\}$ 

```

3.3.5.) The module then aligns (s_i, t_j) pairs with non-zero evidence in decreasing order of this score (lines 13-18). In addition, it aligns all the pairs that contributed contextual evidence for the (s_i, t_j) alignment (lines 19-22). Note that we implement a one-to-one alignment whereby a word gets aligned at most once within the module.

Algorithm 4 (*cwTextAlign*) presents alignment based on similarities in the textual neighborhood. For each potentially alignable pair (s_i, t_j) , Algorithm 2 is used to extract the context, which is a set of neighboring content word pairs (lines 2-7). The contextual similarity is the sum of the similarities of these pairs

Algorithm 4: $cwTextAlign(S, T, A_E, w, STOP)$

Input:

1. S, T : Sentences to be aligned
2. A_E : Existing alignments by word indexes
3. w : Weight of word similarity relative to contextual similarity
4. $STOP$: A set of stop words

Output: $A = \{(i, j)\}$: word index pairs of aligned words $\{(s_i, t_j)\}$ where $s_i \in S$ and $t_j \in T$

```
1  $\Psi \leftarrow \emptyset; \Phi \leftarrow \emptyset$ 
2 for  $s_i \in S, t_j \in T$  do
3   if  $s_i \notin STOP \wedge \neg \exists t_l : (i, l) \in A_E$ 
4      $\wedge t_j \notin STOP \wedge \neg \exists s_k : (k, j) \in A_E$ 
5      $\wedge wordSim(s_i, t_j) > 0$  then
6        $\Psi \leftarrow \Psi \cup \{(i, j)\}$ 
7        $context \leftarrow textContext(S, T, i, j, STOP)$ 
8        $contextSim \leftarrow \sum_{(k, l) \in context} wordSim(s_k, t_l)$ 
9        $\Phi(i, j) \leftarrow w * wordSim(s_i, t_j)$ 
10       $+ (1 - w) * contextSim$ 
11 Linearize and sort  $\Psi$  in decreasing order of  $\Phi(i, j)$ 
12  $A \leftarrow \emptyset$ 
13 for  $(i, j) \in \Psi$  do
14   if  $\neg \exists l : (i, l) \in A$ 
15      $\wedge \neg \exists k : (k, j) \in A$  then
16      $A \leftarrow A \cup \{(i, j)\}$ 
```

(line 8), and the alignment score is a weighted sum of word similarity and contextual similarity (lines 9, 10). The alignment score is then used to make one-to-one word alignment decisions (lines 11-16). Considering textual neighbors as weaker sources of evidence, we do not align the neighbors.

Note that in $cwTextAlign$ we also align semantically similar content word pairs (s_i, t_j) with no contextual similarities if no pairs (s_k, t_j) or (s_i, t_l) exist with a higher alignment score. This is a consequence of our observation of the MSR alignment dev data, where we find that more often than not content words are inherently sufficiently meaningful to be aligned even in the absence of contextual evidence when there are no competing pairs.

The content word alignment module is thus itself a pipeline of $cwDepAlign$ and $cwTextAlign$.

3.3.4 Stop Words

We follow the contextual evidence-based approach to align stop words as well, some of which get aligned

Algorithm 5: $align(S, T, EQ, w, STOP)$

Input:

1. S, T : Sentences to be aligned
2. EQ : Dependency type equivalences (Table 1)
3. w : Weight of word similarity relative to contextual similarity
4. $STOP$: A set of stop words

Output: $A = \{(i, j)\}$: word index pairs of aligned words $\{(s_i, t_j)\}$ where $s_i \in S$ and $t_j \in T$

```
1  $A \leftarrow wsAlign(S, T)$ 
2  $A \leftarrow A \cup neAlign(S, T, EQ, A, w)$ 
3  $A \leftarrow A \cup cwDepAlign(S, T, EQ, A, w, STOP)$ 
4  $A \leftarrow A \cup cwTextAlign(S, T, A, w, STOP)$ 
5  $A \leftarrow A \cup swDepAlign(S, T, A, w, STOP)$ 
6  $A \leftarrow A \cup swTextAlign(S, T, A, w, STOP)$ 
```

as part of word sequence alignment as discussed in Section 3.3.1, and neighbor alignment as discussed in Section 3.3.3. For the rest we utilize dependencies and textual neighborhoods as before, with three adjustments.

Firstly, since stop word alignment is the last module in our pipeline, rather than considering all semantically similar word pairs for contextual similarity, we consider only aligned pairs. Secondly, since many stop words (e.g. determiners, modals) typically demonstrate little variation in the dependencies they engage in, we ignore type equivalences for stop words and implement only exact matching of dependencies. (Stop words in general can participate in equivalent dependencies, but we leave constructing a corresponding mapping for future work.) Finally, for textual neighborhood, we separately check alignments of the left and the right neighbors and aggregate the evidences to determine alignment – again due to the primarily syntactic nature of interaction of stop words with their neighbors.

Thus stop words are also aligned in a sequence of dependency and textual neighborhood-based alignments. We assume two corresponding modules named $swDepAlign$ and $swTextAlign$, respectively.

3.3.5 The Algorithm

Our full alignment pipeline is shown as the method $align$ in Algorithm 5. Note that the strict order of the alignment modules limits the scope of downstream modules since each such module discards any word that has already been aligned by an earlier module

(this is accomplished via the variable A ; the corresponding parameter in Algorithms 3 and 4 is A_E).

The rationales behind the specific order of the modules can now be explained: (1) *wsAlign* is a module with relatively higher precision, (2) it is convenient to align named entities before other content words to enable alignment of entity mentions of different lengths, (3) dependency-based evidence was observed to be more reliable (i.e. of higher precision) than textual evidence in the MSR alignment *dev* set, and (4) stop word alignments are dependent on existing content word alignments.

The aligner assumes two free parameters: *ppdbSim* and w (in Algorithms 3 and 4). To determine their values, we exhaustively search through the two-dimensional space (*ppdbSim*, w) for *ppdbSim*, $w \in \{0.1, \dots, 0.9, 1\}$ and the combination (0.9, 0.9) yields the best F_1 score for the MSR alignment *dev* set. We use these values for the aligner in all of its subsequent applications.

4 Evaluation

We evaluate the performance of our aligner both intrinsically and extrinsically on multiple corpora.

4.1 Intrinsic Evaluation

The MSR alignment dataset² (Brockett, 2007) was designed to train and directly evaluate automated aligners. Three annotators individually aligned words and phrases in 1600 pairs of *premise* and *hypothesis* sentences from the RTE2 challenge data (divided into *dev* and *test* sets, each consisting of 800 sentences). The dataset has subsequently been used to assess several top performing aligners (MacCartney et al., 2008; Thadani and McKeown, 2011; Yao et al., 2013a; Yao et al., 2013b). We use the *test* set for evaluation in the same manner as these studies: (a) we apply majority rule to select from the three sets of annotations for each sentence and discard three-way disagreements, (b) we evaluate only on the *sure* links (word pairs that annotators mentioned should certainly be aligned, as opposed to *possible* links).

We test the generalizability of the aligner by evaluating it, unchanged (i.e. with identical parameter values), on a second alignment corpus: the Edin-

²http://www.cs.biu.ac.il/nlp/files/RTE_2006_Aligned.zip

	System	P%	R%	F ₁ %	E%
MSR	MacCartney et al. (2008)	85.4	85.3	85.3	21.3
	Thadani & McKeown (2011)	89.5	86.2	87.8	33.0
	Yao et al. (2013a)	93.7	84.0	88.6	35.3
	Yao et al. (2013b)	92.1	82.8	86.8	29.1
	This Work	93.7	89.8	91.7	43.8
EDB++	Yao et al. (2013a)	91.3	82.0	86.4	15.0
	Yao et al. (2013b)	90.4	81.9	85.9	13.7
	This Work	93.5	82.5	87.6	18.3

Table 2: Results of intrinsic evaluation on two datasets

burgh++³ (Thadani et al., 2012) corpus. The *test* set consists of 306 pairs; each pair is aligned by at most two annotators and we adopt the random selection policy described in (Thadani et al., 2012) to resolve disagreements.

Table 2 shows the results. For each corpus, it shows *precision* (% alignments that matched with gold annotations), *recall* (% gold alignments discovered by the aligner), F_1 score and the percentage of sentences that received the exact gold alignments (denoted by E) from the aligner.

On the MSR test set, our aligner shows a 3.1% improvement in F_1 score over the previous best system (Yao et al., 2013a) with a 27.2% error reduction. Importantly, it demonstrates a considerable increase in recall without a loss of precision. The E score also increases as a consequence.

On the Edinburgh++ test set, our system achieves a 1.2% increase in F_1 score (an error reduction of 8.8%) over the previous best system (Yao et al., 2013a), with improvements in both precision and recall. This is a remarkable result that demonstrates the general applicability of the aligner, as no parameter tuning took place.

4.1.1 Ablation Test

We perform a set of ablation tests to assess the importance of the aligner’s individual components. Each row of Table 3 beginning with (-) shows a feature excluded from the aligner and two associated sets of metrics, showing the performance of the resulting algorithm on the two alignment corpora.

Without a word similarity module, recall drops as would be expected. Without contextual evidence (word sequences, dependencies and textual neighbors) precision drops considerably and recall also falls. Without dependencies, the aligner still gives

³<http://www.ling.ohio-state.edu/~scott/#edinburgh-plusplus>

Feature	MSR			EDB++		
	P%	R%	F ₁ %	P%	R%	F ₁ %
Original	93.7	89.8	91.7	93.5	82.5	87.6
(-) Word Similarity	95.2	86.3	90.5	95.1	77.3	85.3
(-) Contextual Evidence	81.3	86.0	83.6	86.4	80.6	83.4
(-) Dependencies	94.2	88.8	91.4	93.8	81.3	87.1
(-) Text Neighborhood	85.5	90.4	87.9	90.4	84.3	87.2
(-) Stop Words	94.2	88.3	91.2	92.2	80.0	85.7

Table 3: Ablation test results

state-of-the-art results, which points to the possibility of a very fast yet high-performance aligner. Without evidence from textual neighbors, however, the precision of the aligner suffers badly. Textual neighbors find alignments across different lexical categories, a type of alignment that is currently not supported by our dependency equivalences. Extending the set of dependency equivalences might alleviate this issue. Finally, without stop words (i.e. while aligning content words only) recall drops just a little for each corpus, which is expected as content words form the vast majority of non-identical word alignments.

4.2 Extrinsic Evaluation

We extrinsically evaluate our system on textual similarity identification and paraphrase detection. Here we discuss each task and the results of evaluation.

4.2.1 Semantic Textual Similarity

Given two short input text fragments (commonly sentences), the goal of this task is to identify the degree to which the two fragments are semantically similar. The *SEM 2013 STS task (Agirre et al., 2013) assessed a number of STS systems on four test datasets by comparing their output scores to human annotations. Pearson correlation coefficient with human annotations was computed individually for each test set and a weighted sum of the correlations was used as the final evaluation metric (the weight for each dataset was proportional to its size).

We apply our aligner to the task by aligning each sentence pair and taking the proportion of content words aligned in the two sentences (by normalizing with the harmonic mean of their number of content words) as a proxy of their semantic similarity. Only three of the four STS 2013 datasets are freely available⁴ (*headlines*, *OnWN*, and *FNWN*), which we use for our experiments (leaving out the *SMT* dataset).

⁴<http://ixa2.si.ehu.es/sts/>

System	Correl.%	Rank
Han et al. (2013)	73.7	1 (original)
JacanaAlign	46.2	66
This Work	67.2	7

Table 4: Extrinsic evaluation on STS 2013 data

These three sets contain 1500 annotated sentence pairs in total.

Table 4 shows the results. The first row shows the performance of the top system in the task. With a direct application of our aligner (no parameter tuning), our STS algorithm achieves a 67.15% weighted correlation, which would earn it the 7th rank among 90 participating systems. Considering the fact that alignment is one of many components of STS, this result is truly promising.

For comparison, we also evaluate the previous best aligner named JacanaAlign (Yao et al., 2013a) on STS 2013 data (the JacanaAlign public release⁵ is used, which is a version of the original aligner with extra lexical resources). We apply three different values derived from its output as proxies of semantic similarity: a) aligned content word proportion, b) the Viterbi decoding score, and c) the normalized decoding score. Of the three, (b) gives the best results, which we show in row 2 of Table 4. Our aligner outperforms JacanaAlign by a large margin.

4.2.2 Paraphrase Identification

The goal of paraphrase identification is to decide if two sentences have the same meaning. The output is a yes/no decision instead of a real-valued similarity score as in STS. We use the MSR paraphrase corpus⁶ (4076 *dev* pairs, 1725 *test* pairs) (Dolan et al., 2004) to evaluate our aligner and compare with other aligners. Following earlier work (MacCartney et al., 2008; Yao et al., 2013b), we use a normalized alignment score of the two sentences to make a decision based on a threshold which we set using the *dev* set. Alignments with a higher-than-threshold score are taken to be paraphrases and the rest non-paraphrases.

Again, this is an oversimplified application of the aligner, even more so than in STS, since a small change in linguistic properties of two sentences (e.g. polarity or modality) can turn them into non-

⁵<https://code.google.com/p/jacana/>

⁶<http://research.microsoft.com/en-us/downloads/607d14d9-20cd-47e3-85bc-a2f65cd28042/>

System	Acc.%	P%	R%	F ₁ %
Madnani et al. (2012)	77.4	79.0	89.9	84.1
Yao et al. (2013a)	70.0	72.6	88.1	79.6
Yao et al. (2013b)	68.1	68.6	95.8	79.9
This Work	73.4	76.6	86.4	81.2

Table 5: Extrinsic evaluation on MSR paraphrase data

paraphrases despite having a high degree of alignment. So the aligner was not expected to demonstrate state-of-the-art performance, but still it gets close as shown in Table 5. The first column shows the accuracy of each system in classifying the input sentences into one of two classes: *true* (paraphrases) and *false* (non-paraphrases). The rest of the columns show the performance of the system for the true class in terms of precision, recall, and F₁ score. *Italicized* numbers represent scores that were not reported by the authors of the corresponding papers, but have been reconstructed from the reported data (and hence are likely to have small precision errors).

The first row shows the best performance by any system on the test set to the best of our knowledge. The next two rows show the performances of two state-of-the-art aligners (performances of both systems were reported in (Yao et al., 2013b)). The last row shows the performance of our aligner. Although it does worse than the best paraphrase system, it outperforms the other aligners.

5 Discussion

Our experiments reveal that a word aligner based on simple measures of lexical and contextual similarity can demonstrate state-of-the-art accuracy. However, as alignment is frequently a component of larger tasks, high accuracy alone is not always sufficient. Other dimensions of an aligner’s usability include speed, consumption of computing resources, replicability, and generalizability to different applications. Our design goals include achieving a balance among such multifarious and conflicting goals.

A speed advantage of our aligner stems from formulating the problem as one-to-one word alignment and thus avoiding an expensive decoding phase. The presence of multiple phases is offset by discarding already aligned words in subsequent phases. The use of PPDB as the only (hashable) word similarity resource helps in reducing latency as well as space requirements. As shown in Section 4.1.1, further

speedup could be achieved with only a small performance degradation by considering only the textual neighborhood as source of contextual evidence.

However, the two major goals that we believe the aligner achieves to the greatest extent are replicability and generalizability. The easy replicability of the aligner stems from its use of only basic and frequently used NLP modules (a lemmatizer, a POS tagger, an NER module, and a dependency parser: all available as part of the Stanford CoreNLP suite⁷; we use a Python wrapper⁸) and a single word similarity resource (PPDB).

We experimentally show that the aligner can be successfully applied to different alignment datasets as well as multiple end tasks. We believe a design characteristic that enhances the generalizability of the aligner is its minimal dependence on the MSR alignment training data, which originates from a textual entailment corpus having unique properties such as disparities in the lengths of the input sentences and a directional nature of their relationship (i.e., the premise implying the hypothesis, but not vice versa). A related potential reason is the symmetry of the aligner’s output (caused by its assumption of no directionality) – the fact that it outputs the same set of alignments regardless of the order of the input sentences, in contrast to most existing aligners.

Major limitations of the aligner include the inability to align phrases, including multiword expressions. It is incapable of capturing and exploiting long distance dependencies among words (e.g. coreferences). No word similarity resource is perfect and PPDB is no exception, therefore certain word alignments also remain undetected.

6 Conclusions

We show how contextual evidence can be used to construct a monolingual word aligner with certain desired properties, including state-of-the-art accuracy, easy replicability, and high generalizability. Some potential avenues for future work include: allowing phrase-level alignment via phrasal similarity resources (e.g. the phrasal paraphrases of PPDB), including other sources of similarity such as vector space models or WordNet relations, expanding the set

⁷<http://nlp.stanford.edu/downloads/corenlp.shtml>

⁸<https://github.com/dasmith/stanford-corenlp-python>

of dependency equivalences and/or using semantic role equivalences, and formulating our alignment algorithm as objective optimization rather than greedy search.

The aligner is available for download at <https://github.com/ma-sultan/monolingual-word-aligner>.

Acknowledgments

This material is based in part upon work supported by the National Science Foundation under Grant Numbers EHR/0835393 and EHR/0835381. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

- Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic Textual Similarity. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics, 32-43.
- Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with Bilingual Parallel Corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 597-604.
- Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch. 2012. UKP: computing semantic textual similarity by combining multiple content similarity measures. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics, 435-440.
- Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The Second PASCAL Recognising Textual Entailment Challenge. In *Proceedings of The Second PASCAL Recognising Textual Entailment Challenge*.
- Chris Brockett. 2007. Aligning the RTE 2006 Corpus. Technical Report MSR-TR-2007-77, Microsoft Research.
- Nathanael Chambers, Daniel Cer, Trond Grenager, David Hall, Chloe Kiddon, Bill MacCartney, Marie-Catherine de Marneffe, Daniel Ramage, Eric Yeh, and Christopher D Manning. 2007. Learning alignments and leveraging natural logic. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*. Association for Computational Linguistics, 165-170.
- Ming-Wei Chang, Dan Goldwasser, Dan Roth, and Vivek Srikumar. 2010. Discriminative Learning over Constrained Latent Representations. In *Proceedings of the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 429-437.
- Dipanjan Das and Noah A. Smith. 2009. Paraphrase Identification as Probabilistic Quasi-Synchronous Recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Association for Computational Linguistics, 468-476.
- Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of the International Conference on Language Resources and Evaluation*. 449-454.
- Marie-Catherine de Marneffe and Christopher D. Manning. 2008. Stanford typed dependencies manual. Technical Report, Stanford University.
- Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources. In *Proceedings of the International Conference on Computational Linguistics*. Association for Computational Linguistics, 350-356.
- Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 363-370.
- Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 758-764.
- Lushan Han, Abhay Kashyap, Tim Finin, James Mayeld, and Jonathan Weese. 2013. UMBC EBIQUITY-CORE: Semantic Textual Similarity Systems. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics, Volume 1*. Association for Computational Linguistics, 44-52.
- Andrew Hickl and Jeremy Bensley. 2007. A Discourse Commitment-Based Framework for Recognizing Textual Entailment. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*. Association for Computational Linguistics, 171-176.
- Andrew Hickl, Jeremy Bensley, John Williams, Kirk Roberts, Bryan Rink, and Ying Shi. 2006. Recognizing Textual Entailment with LCCs GROUNDHOG

- System. In *Proceedings of the Second PASCAL Challenges Workshop on Recognizing Textual Entailment*.
- Milen Kouylekov and Bernardo Magnini. 2005. Recognizing textual entailment with tree edit distance algorithms. In *Proceedings of the PASCAL Challenges Workshop: Recognising Textual Entailment Challenge* 17-20.
- Bill MacCartney, Michel Galley, and Christopher D. Manning. 2008. A Phrase-Based Alignment Model for Natural Language Inference. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 802-811.
- Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining Machine Translation Metrics for Paraphrase Identification. In *Proceedings of 2012 Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 182-190.
- Kapil Thadani and Kathleen McKeown. 2011. Optimal and Syntactically-Informed Decoding for Monolingual Phrase-Based Alignment. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 254-259.
- Kapil Thadani, Scott Martin, and Michael White. 2012. A Joint Phrasal and Dependency Model for Paraphrase Alignment. In *Proceedings of COLING 2012: Posters*. 1229-1238.
- 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 the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 173-180.
- Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013a. A Lightweight and High Performance Monolingual Word Aligner. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 702-707.
- Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013b. Semi-Markov Phrase-based Monolingual Alignment. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 590-600.
- Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013c. Answer Extraction as Sequence Tagging with Tree Edit Distance. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 858-867.

From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions

Peter Young Alice Lai Micah Hodosh Julia Hockenmaier

Department of Computer Science

University of Illinois at Urbana-Champaign

{pyoung2, aylai2, mhodosh2, juliahmr}@illinois.edu

Abstract

We propose to use the *visual denotations* of linguistic expressions (i.e. the set of images they describe) to define novel *denotational similarity metrics*, which we show to be at least as beneficial as distributional similarities for two tasks that require semantic inference. To compute these denotational similarities, we construct a *denotation graph*, i.e. a subsumption hierarchy over constituents and their denotations, based on a large corpus of 30K images and 150K descriptive captions.

1 Introduction

The ability to draw inferences from text is a prerequisite for language understanding. These inferences are what makes it possible for even brief descriptions of everyday scenes to evoke rich mental images. For example, we would expect an image of *people shopping in a supermarket* to depict aisles of produce or other goods, and we would expect most of these people to be customers who are either standing or walking around. But such inferences require a great deal of commonsense world knowledge. Standard *distributional* approaches to lexical similarity (Section 2.1) are very effective at identifying which words are related to the same topic, and can provide useful features for systems that perform semantic inferences (Mirkin et al., 2009), but are not suited to capture precise entailments between complex expressions. In this paper, we propose a novel approach for the automatic acquisition of *denotational* similarities between descriptions of everyday situations (Section 2). We define the (*visual*)

denotation of a linguistic expression as the set of images it describes. We create a corpus of images of everyday activities (each paired with multiple captions; Section 3) to construct a large scale *visual denotation graph* which associates image descriptions with their denotations (Section 4). The algorithm that constructs the denotation graph uses purely syntactic and lexical rules to produce simpler captions (which have a larger denotation). But since each image is originally associated with several captions, the graph can also capture similarities between syntactically and lexically unrelated descriptions. We apply these similarities to two different tasks (Sections 6 and 7): an approximate entailment recognition task for our domain, where the goal is to decide whether the hypothesis (a brief image caption) refers to the same image as the premises (four longer captions), and the recently introduced Semantic Textual Similarity task (Agirre et al., 2012), which can be viewed as a graded (rather than binary) version of paraphrase detection. Both tasks require semantic inference, and our results indicate that denotational similarities are at least as effective as standard approaches to similarity. Our code and data set, as well as the denotation graph itself and the lexical similarities we define over it are available for research purposes at <http://nlp.cs.illinois.edu/Denotation.html>.

2 Towards Denotational Similarities

2.1 Distributional Similarities

The distributional hypothesis posits that linguistic expressions that appear in similar contexts have a



Gray haired man in black suit and yellow tie working in a financial environment.
A graying man in a suit is perplexed at a business meeting.
A businessman in a yellow tie gives a frustrated look.
A man in a yellow tie is rubbing the back of his neck.
A man with a yellow tie looks concerned.



A butcher cutting an animal to sell.
A green-shirted man with a butcher's apron uses a knife to carve out the hanging carcass of a cow.
A man at work, butchering a cow.
A man in a green t-shirt and long tan apron hacks apart the carcass of a cow while another man hoses away the blood.
Two men work in a butcher shop; one cuts the meat from a butchered cow, while the other hoses the floor.

Figure 1: Two images from our data set and their five captions

similar meaning (Harris, 1954). This has led to the definition of vector-based distributional similarities, which represent each word w as a vector \mathbf{w} derived from counts of w 's co-occurrence with other words. These vectors can be used directly to compute the *lexical similarities* of words, either via the cosine of the angle between them, or via other, more complex metrics (Lin, 1998). More recently, asymmetric similarities have been proposed as more suitable for semantic inference tasks such as entailment (Weeds and Weir, 2003; Szpektor and Dagan, 2008; Clarke, 2009; Kotlerman et al., 2010). Distributional word vectors can also be used to define the *compositional similarity* of longer strings (Mitchell and Lapata, 2010). To compute the similarity of two strings, the lexical vectors of the words in each string are first combined into a single vector (e.g. by element-wise addition or multiplication), and then an appropriate vector similarity (e.g. cosine) is applied to the resulting pair of vectors.

2.2 Visual Denotations

Our approach is inspired by truth-conditional semantic theories in which the *denotation* of a declarative sentence is assumed to be the set of all situations or possible worlds in which the sentence is true (Montague, 1974; Dowty et al., 1981; Barwise and Perry, 1980). Restricting our attention to visually descriptive sentences, i.e. non-negative, episodic (Carlson, 2005) sentences that can be used to describe an image (Figure 1), we propose to instantiate the abstract notions of possible worlds or situations with concrete sets of images. The interpretation function $\llbracket \cdot \rrbracket$ maps sentences to their **visual denotations** $\llbracket s \rrbracket$, which is the set of images $\mathbf{i} \in U_s \subseteq U$ in

a ‘universe’ of images U that s describes:

$$\llbracket s \rrbracket = \{ \mathbf{i} \in U \mid s \text{ is a truthful description of } \mathbf{i} \} \quad (1)$$

Similarly, we map nouns and noun phrases to the set of images that depict the objects they describe, and verbs and verb phrases to the set of images that depict the events they describe.

2.3 Denotation Graphs

Denotations induce a partial ordering over descriptions: if s (e.g. “*a poodle runs on the beach*”) entails a description s' (e.g. “*a dog runs*”), its denotation is a subset of the denotation of s' ($\llbracket s \rrbracket \subseteq \llbracket s' \rrbracket$), and we say that s' *subsumes* the more specific s ($s' \sqsubseteq s$). In our domain of descriptive sentences, we can obtain more generic descriptions by simple **syntactic and lexical operations** $\omega \in O \subset S \times S$ that preserve upward entailment, so that if $\omega(s) = s'$, $\llbracket s \rrbracket \subseteq \llbracket s' \rrbracket$. We consider three types of operations: the removal of optional material (e.g PPs like *on the beach*), the extraction of simpler constituents (NPs, VPs, or simple Ss), and lexical substitutions of nouns by their hypernyms (*poodle* \rightarrow *dog*). These operations are akin to the *atomic edits* of MacCartney and Manning (2008)'s NatLog system, and allow us to construct large subsumption hierarchies over image descriptions, which we call **denotation graphs**. Given a set of (upward entailment-preserving) operations $O \subset S \times S$, the denotation graph $DG = \langle E, V \rangle$ of a set of images I and a set of strings S represents a subsumption hierarchy in which each node $V = \langle s, \llbracket s \rrbracket \rangle$ corresponds to a string $s \in S$ and its denotation $\llbracket s \rrbracket \subseteq I$. Directed edges $e = (s, s') \in E \subseteq V \times V$ indicate a subsumption relation $s \sqsubseteq s'$ between a more generic expression s and its child s' . An edge from s to s'

exists if there is an operation $\omega \in O$ that *reduces* the string s' to s (i.e. $\omega(s') = s$) and its inverse ω^{-1} *expands* the string s to s' (i.e. $\omega^{-1}(s) = s'$).

2.4 Denotational Similarities

Given a denotation graph over N images, we estimate the denotational probability of an expression s with a denotation of size $|\llbracket s \rrbracket|$ as $P_{\llbracket \cdot \rrbracket}(s) = |\llbracket s \rrbracket|/N$, and the joint probability of two expressions analogously as $P_{\llbracket \cdot \rrbracket}(s, s') = |\llbracket s \rrbracket \cap \llbracket s' \rrbracket|/N$. The conditional probability $P_{\llbracket \cdot \rrbracket}(s | s')$ indicates how likely s is to be true when s' holds, and yields a simple directed denotational similarity. The (normalized) pointwise mutual information (PMI) (Church and Hanks, 1990) defines a symmetric similarity:

$$nPMI_{\llbracket \cdot \rrbracket}(s, s') = \frac{\log\left(\frac{P_{\llbracket \cdot \rrbracket}(s, s')}{P_{\llbracket \cdot \rrbracket}(s)P_{\llbracket \cdot \rrbracket}(s')}\right)}{-\log(P_{\llbracket \cdot \rrbracket}(s, s'))}$$

We set $P_{\llbracket \cdot \rrbracket}(s|s) = nPMI_{\llbracket \cdot \rrbracket}(s, s) = 1$, and, if s or s' are not in the denotation graph, $nPMI_{\llbracket \cdot \rrbracket}(s, s') = P_{\llbracket \cdot \rrbracket}(s, s') = 0$.

3 Our Data Set

Our data set (Figure 1) consists of 31,783 photographs of everyday activities, events and scenes (all harvested from Flickr) and 158,915 captions (obtained via crowdsourcing). It contains and extends Hodosh et al. (2013)’s corpus of 8,092 images. We followed Hodosh et al. (2013)’s approach to collect images. We also use their annotation guidelines, and use similar quality controls to correct spelling mistakes, eliminate ungrammatical or non-descriptive sentences. Almost all of the images that we add to those collected by Hodosh et al. (2013) have been made available under a Creative Commons license. Each image is described independently by five annotators who are not familiar with the specific entities and circumstances depicted in them, resulting in captions such as “*Three people setting up a tent*”, rather than the kind of captions people provide for their own images (“*Our trip to the Olympic Peninsula*”). Moreover, different annotators use different levels of specificity, from describing the overall situation (*performing a musical piece*) to specific actions (*bowing on a violin*). This variety of descriptions associated with the same image is what allows us to induce denotational similari-

ties between expressions that are not trivially related by syntactic rewrite rules.

4 Constructing the Denotation Graph

The construction of the denotation graph consists of the following steps: preprocessing and linguistic analysis of the captions, identification of applicable transformations, and generation of the graph itself.

Preprocessing and Linguistic Analysis We use the Linux spell checker, the OpenNLP tokenizer, POS tagger and chunker (<http://opennlp.apache.org>), and the Malt parser (Nivre et al., 2006) to analyze the captions. Since the vocabulary of our corpus differs significantly from the data these tools are trained on, we resort to a number of heuristics to improve the analyses they provide. Since some heuristics require us to identify different entity types, we developed a lexicon of the most common entity types in our domain (people, clothing, bodily appearance (e.g. hair or body parts), containers of liquids, food items and vehicles).

After spell-checking, we normalize certain words and compounds with several spelling variations, e.g. *barbecue* (*barbeque*, *BBQ*), *gray* (*grey*), *waterski* (*water ski*), *brown-haired* (*brown haired*), and tokenize the captions using the OpenNLP tokenizer. The OpenNLP POS tagger makes a number of systematic errors on our corpus (e.g. mistagging main verbs as nouns). Since these errors are highly systematic, we are able to correct them automatically by applying deterministic rules (e.g. *climbs* is never a noun in our corpus, *stand* is a noun if it is preceded by *vegetable* but a verb when preceded by a noun that refers to people). These fixes apply to 27,784 (17% of the 158,915 image captions). Next, we use the OpenNLP chunker to create a shallow parse. Fixing its (systematic) errors affects 28,587 captions. We then analyze the structure of each NP chunk to identify heads, determiners and prenominal modifiers. The head may include more than a single token if WordNet (or our hypernym lexicon, described below) contains a corresponding entry (e.g. *little girl*). Determiners include phrases such as *a couple* or *a few*. Although we use the Malt parser (Nivre et al., 2006) to identify subject-verb-object dependencies, we have found it more accurate to develop deterministic heuristics and lexi-

cal rules to identify the boundaries of complex (e.g. conjoined) NPs, allowing us to treat “*a man with red shoes and a white hat*” as an NP followed by a single PP, but “*a man with red shoes and a white-haired woman*” as two NPs, and to transform e.g. “*standing by a man and a woman*” into “*standing*” and not “*standing and a woman*” when dropping the PP.

Hypernym Lexicon We use our corpus and WordNet to construct a hypernym lexicon that allows us to replace head nouns with more generic terms. We only consider hypernyms that occur themselves with sufficient frequency in the original captions (replacing “*adult*” with “*person*”, but not with “*organism*”). Since the language in our corpus is very concrete, each noun tends to have a single sense, allowing us to always replace it with the same hypernyms.¹ But since WordNet provides us with multiple senses for most nouns, we first have to identify which sense is used in our corpus. To do this, we use the heuristic cross-caption coreference algorithm of Hodosh et al. (2010) to identify coreferent NP chunks among the original five captions of each image.² For each ambiguous head noun, we consider every non-singleton coreference chains it appears in, and reduce its synsets to those that stand in a hypernym-hyponym relation with at least one other head noun in the chain. Finally, we apply a greedy majority voting algorithm to iteratively narrow down each term’s senses to a single synset that is compatible with the largest number of coreference chains it occurs in.

Caption Normalization In order to increase the recall of the denotations we capture, we drop all punctuation marks, and lemmatize nouns, verbs, and adjectives that end in “*-ed*” or “*-ing*” before gener-

¹Descriptions of people that refer to both age and gender (e.g. “*man*”) can have multiple distinct hypernyms (“*adult*”/“*male*”). Because our annotators never describe young children or babies as “*persons*”, we only allow terms that are likely to describe adults or teenagers (including occupations) to be replaced by the term “*person*”. This means that the term “*girl*” has two senses: a female child (the default) or a younger woman. We distinguish the two senses in a preprocessing step: if the other captions of the same image do not mention children, but refer to teenaged or adult women, we assign *girl* the *woman*-sense. Some nouns that end in *-er* (e.g. “*diner*”, “*pitcher*”) also violate our monosemy assumption.

²Coreference resolution has also been used for word sense disambiguation by Preiss (2001) and Hu and Liu (2011).

ating the denotation graph. In order to distinguish between frequently occurring homonyms where the noun is unrelated to the verb, we change all forms of the verb *dress* to *dressed*, all forms of the verb *stand* to *standing* and all forms of the verb *park* to *parking*. Finally, we drop sentence-initial *there/here/this is/are* (as in *there is a dog splashing in the water*), and normalize the expressions *in X* and *dressed (up) in X* (where *X* is an article of clothing or a color) to *wear X*. We reduce plural determiners to {*two, three, some*}, and drop singular determiners except for *no*.

4.1 Rule Templates

The denotation graph contains a directed edge from *s* to *s'* if there is a rule ω that reduces *s'* to *s*, with an inverse ω^{-1} that expands *s* to *s'*. Reduction rules can drop optional material, extract simpler constituents, or perform lexical substitutions.

Drop Pre-Nominal Modifiers: “*red shirt*” → “*shirt*” In an NP of the form “*X Y Z*”, where *X* and *Y* both modify the head *Z*, we only allow *X* and *Y* to be dropped separately if “*X Z*” and “*Y Z*” both occur elsewhere in the corpus. Since “*white building*” and “*stone building*” occur elsewhere in the corpus, we generate both “*white building*” and “*stone building*” from the NP “*white stone building*”. But since “*ice player*” is not used, we replace “*ice hockey player*” only with “*hockey player*” (which does occur) and then “*player*”.

Drop Other Modifiers “*run quickly*” → “*run*” We drop ADVP chunks and adverbs in VP chunks. We also allow a prepositional phrase (a preposition followed by a possibly conjoined NP chunk) to be dropped if the preposition is locational (“*in*”, “*on*”, “*above*”, etc.), directional (“*towards*”, “*through*”, “*across*”, etc.), or instrumental (“*by*”, “*for*”, “*with*”). Similarly, we also allow the dropping of all “*wear NP*” constructions. Since the distinction between particles and prepositions is often difficult, we also use a predefined list of phrasal verbs that commonly occur in our corpus to identify constructions such as “*climb up a mountain*”, which is transformed into “*climb a mountain*” or “*walk down a street*”, which is transformed into “*walk*”.

Replace Nouns by Hypernyms: “*red shirt*” → “*red clothing*” We iteratively use our hypernym


```

GENERATEGRAPH():
Q, Captions, Rules ← ∅
for all  $c \in \text{ImageCorpus}$  do
  Rules( $c$ ) ← GenerateRules( $s_c$ )
  pushAll(Q, { $c$ } × RootNodes( $s_c$ , Rules( $c$ )))
while ¬empty(Q) do
  ( $c$ ,  $s$ ) ← pop(Q)
  Captions( $s$ ) ← Captions( $s$ ) ∪ { $c$ }
  if |Captions( $s$ )| = 2 then
    for all  $c' \in \text{Captions}(s)$  do
      pushAll(Q, { $c'$ } × Children( $s$ , Rules( $c'$ )))
  else if |Captions( $s$ )| > 2 then
    pushAll(Q, { $c$ } × Children( $s$ , Rules( $c$ )))
  Figure 2: Generating the graph

```

lexicon to make head nouns more generic. We only allow head nouns to be replaced by their hypernyms if any age based modifiers have already been removed: “*toddler*” can be replaced with “*child*”, but not “*older toddler*” with “*older child*”.

Handle Partitive NPs: *cup of tea* → “*cup*”, “*tea*”
 In most partitive NP₁-of-NP₂ constructions (“*cup of tea*”, “*a team of football players*”) the corresponding entity can be referred to by both the first or the second NP. Exceptions include the phrase “*body of water*”, and expressions such as “*a kind/type/sort of*”, which we treat similar to determiners.

Handle VP₁-to-VP₂ Cases Depending on the first verb, we replace VPs of the form **X to Y** with both **X** and **Y** if **X** is a movement or posture (*jump to catch*, etc.). Otherwise we distinguish between cases we can only replace with **X** (*wait to jump*) and those we can only replace with **Y** (*seem to jump*).

Extract Simpler Constituents Any noun phrase or verb phrase can also be used as a node in the graph and simplified further. We use the Malt dependencies (and the person terms in the entity type lexicon) to identify and extract subject-verb-object chunks which correspond to simpler sentences that we would otherwise not be able to obtain: from “*man laugh(s) while drink(ing)*”, we extract “*man laugh*” and “*man drink*”, and then further split those into “*man*”, “*laugh(s)*”, and “*drink*”.

4.2 Graph Generation

The naive approach to graph generation would be to generate all possible strings for each caption. However, this would produce far more strings than can be

processed in a reasonable amount of time, and most of these strings would have uninformative denotations, consisting of only a single image. To make graph generation tractable, we use a top-down algorithm which generates the graph from the most generic (root) nodes, and stops at nodes that have a singleton denotation (Figure 2). We first identify the set of rules that can apply to each original caption (*GenerateRules*). These rules are then used to reduce each caption as much as possible. The resulting (maximally generic) strings are added as root nodes to the graph (*RootNodes*), and added to the queue Q. Q keeps track of all currently possible node expansions. It contains items $\langle c, s \rangle$, which pair the ID of an original caption and its image (c) with a string (s) that corresponds to an existing node in the graph and can be derived from c ’s caption. When $\langle c, s \rangle$ is processed, we check how many captions have generated s so far (Captions(s)). If s has more than a single caption, we use each of the applicable rewrite rules of c ’s caption to create new strings s' that correspond to the children of s in the graph, and push all resulting $\langle c, s' \rangle$ onto Q. If c is the second caption of s , we also use all of the applicable rewrite rules from the first caption c' to create its children.

A post-processing step (not shown in Figure 2) attaches each original caption to all leaf nodes of the graph to which it can be reduced. Finally, we obtain the denotation of each node s from the set of images whose captions are in Captions(s).

5 The Denotation Graph

Size and Coverage On our corpus of 158,439 unique captions and 31,783 images, the denotation graph contains 1,749,097 captions, out of which 230,811 describe more than a single image. Table 1 provides the distribution of the size of denotations. It is perhaps surprising that the 161 captions which describe each over 1,000 images do not just consist of nouns such as *person*, but also contain simple sentences such as *woman standing*, *adult work*, *person walk street*, or *person play instrument*. Since the graph is derived from the original captions by very simple syntactic operations, the denotations it captures are most likely incomplete: $\llbracket \text{soccer player} \rrbracket$ contains 251 images, $\llbracket \text{play soccer} \rrbracket$ contains 234 images, and $\llbracket \text{soccer game} \rrbracket$ contains

Size of denotations	$ \llbracket s \rrbracket \geq 1$	$ \llbracket s \rrbracket \geq 2$	$ \llbracket s \rrbracket \geq 5$	$ \llbracket s \rrbracket \geq 10$	$ \llbracket s \rrbracket \geq 100$	$ \llbracket s \rrbracket \geq 1000$
Nr. of captions	1,749,096	230,811	53,341	22,683	1,921	161

Table 1: Distribution of the size of denotations in our graph

119 images. We have not yet attempted to identify variants in word order (“*stick tongue out*” vs. “*stick out tongue*”) or equivalent choices of preposition (“*look into mirror*” vs. “*look in mirror*”). Despite this brittleness, the current graph already gives us a large number of semantic associations.

Denotational Similarities The following examples of the similarities found by $nPMI_{\llbracket \cdot \rrbracket}$ and $P_{\llbracket \cdot \rrbracket}$ show that denotational similarities do not simply find topically related events, but instead find events that are related by entailment:

$P_{\llbracket \cdot \rrbracket}(x y)$	x	y
0.962	<i>sit</i>	<i>eat lunch</i>
0.846	<i>play guitar</i>	<i>strum</i>
0.811	<i>surf</i>	<i>catch wave</i>
0.800	<i>ride horse</i>	<i>rope calf</i>
0.700	<i>listen</i>	<i>sit in classroom</i>

If someone is *eating lunch*, it is likely that they are *sitting*, and people who *sit in a classroom* are likely to be *listening* to somebody. These entailments can be very precise: “*walk up stair*” entails “*ascend*”, but not “*descend*”; the reverse is true for “*walk down stair*”:

$P_{\llbracket \cdot \rrbracket}(x y)$	$x = \text{ascend}$	$x = \text{descend}$
$y = \text{walk up stair}$	32.0	0.0
$y = \text{walk down stair}$	0.0	30.8

$nPMI_{\llbracket \cdot \rrbracket}$ captures paraphrases as well as closely related events: people *look in a mirror* when *shaving their face*, and baseball players may *try to tag* someone who is *sliding into base*:

$nPMI_{\llbracket \cdot \rrbracket}$	x	y
0.835	<i>open present</i>	<i>unwrap</i>
0.826	<i>lasso</i>	<i>try to rope</i>
0.791	<i>get ready to kick</i>	<i>run towards ball</i>
0.785	<i>try to tag</i>	<i>slide into base</i>
0.777	<i>shave face</i>	<i>look in mirror</i>

Comparing the expressions that are most similar to “*play baseball*” or “*play football*” according to the denotational $nPMI_{\llbracket \cdot \rrbracket}$ and the compositional Σ similarities reveals that the denotational similarity finds a number of actions that are part of the particular sport, while the compositional similarity finds events that are similar to *playing baseball (football)*:

play baseball			
$nPMI_{\llbracket \cdot \rrbracket}$		Σ	
0.674	<i>tag him</i>	0.859	<i>play softball</i>
0.637	<i>hold bat</i>	0.782	<i>play game</i>
0.616	<i>try to tag</i>	0.768	<i>play ball</i>
0.569	<i>slide into base</i>	0.741	<i>play catch</i>
0.516	<i>pitch ball</i>	0.739	<i>play cricket</i>
play football			
$nPMI_{\llbracket \cdot \rrbracket}$		Σ	
0.623	<i>tackle person</i>	0.826	<i>play game</i>
0.597	<i>hold football</i>	0.817	<i>play rugby</i>
0.545	<i>run down field</i>	0.811	<i>play soccer</i>
0.519	<i>wear white jersey</i>	0.796	<i>play on field</i>
0.487	<i>avoid</i>	0.773	<i>play ball</i>

6 Task 1: Approximate Entailment

A caption never provides a complete description of the depicted scene, but commonsense knowledge often allows us to draw implicit inferences: when somebody mentions a *bride*, it is quite likely that the picture shows a woman in a *wedding dress*; a picture of a *parent* most likely also has a *child* or *baby*, etc. In order to compare the utility of denotational and distributional similarities for drawing these inferences, we apply them to an *approximate entailment task*, which is loosely modeled after the Recognizing Textual Entailment problem (Dagan et al., 2006), and consists of deciding whether a brief caption \mathbf{h} (the hypothesis) can describe the same image as a set of captions $\mathbf{P} = \{\mathbf{p}_1, \dots, \mathbf{p}_N\}$ known to describe the same image (the premises).

Data We generate positive and negative items $\langle \mathbf{P}, \mathbf{h}, \pm \rangle$ (Figure 3) as follows: Given an image, any subset of four of its captions form a set of premises. A hypothesis is either a short verb phrase or sentence that corresponds to a node in the denotation graph. By focusing on short hypotheses, we minimize the possibility that they contain extraneous details that cannot be inferred from the premises. Positive examples are generated by choosing a node \mathbf{h} as hypothesis and an image $i \in \llbracket \mathbf{h} \rrbracket$ such that exactly one caption of i generates \mathbf{h} and the other four captions of i are not descendants of \mathbf{h} and hence do not trivially entail \mathbf{h} , giving an unfair advantage to denotational approaches. Negative examples are generated by choosing a node \mathbf{h} as hypothesis and selecting four of the captions of an image $i \notin \llbracket \mathbf{h} \rrbracket$.

Premises:	A woman with dark hair in bending, open mouthed, towards the back of a dark headed toddler’s head. A dark-haired woman has her mouth open and is hugging a little girl while sitting on a red blanket. A grown lady is snuggling on the couch with a young girl and the lady has a frightened look. A mom holding her child on a red sofa while they are both having fun.
VP Hypothesis:	make face
Premises:	A man editing a black and white photo at a computer with a pencil in his ear. A man in a white shirt is working at a computer. A guy in white t-shirt on a mac computer. A young main is using an Apple computer.
S Hypothesis:	man sit

Figure 3: Positive examples from the Approximate Entailment tasks.

Since our items are created automatically, a positive hypothesis is not necessarily logically entailed by its premises. We have performed a small-scale human evaluation on 300 items (200 positive, 100 negative), each judged independently by the same three judges (inter-annotator agreement: Fleiss- κ = 0.74). Our results indicate that over half (55%) of the positive hypotheses can be inferred from their premises alone without looking at the original image, while almost none of the negative hypotheses (100% for sentences, 96% for verb phrases) can be inferred from their premises. The training items are generated from the captions of 25,000 images, and the test items are generated from a disjoint set of 3,000 images. The VP data set consists of 290,000 training items and 16,000 test items, while the S data set consists of 400,000 training items and 22,000 test items. Half of the items in each set are positive, and the other half are negative.

Models All of our models are binary MaxEnt classifiers, trained using MALLET (McCallum, 2002). We have two baseline models: a plain bag-of-words model (BOW) and a bag-of-words model where we add all hypernyms in our lexicon to the captions before computing their overlap (BOW-H). This is intended to minimize the advantage the denotational features obtain from the hypernym lexicon used to construct the denotation graph. In both cases, a global BOW feature captures the fraction of tokens in the hypothesis that are contained in the premises. Word-specific BOW features capture the product of the frequencies of each word in \mathbf{h} and \mathbf{P} . All other models extend the BOW-H model.

Denotational Similarity Features We compute denotational similarities $nPMI_{\square}$ and P_{\square} (Sec-

tion 2.4) over the pairs of nodes in a denotation graph that is restricted to the training images. We only consider pairs of nodes \mathbf{n}, \mathbf{n}' if their denotations contain at least 10 images and their intersection contains at least 2 images.

To map an item $\langle \mathbf{P}, \mathbf{h} \rangle$ to denotational similarity features, we represent the premises as the set of all nodes P that are ancestors of its captions. A sentential hypothesis is represented as the set of nodes $H = \{h_S, h_{sbj}, h_{VP}, h_v, h_{dobj}\}$ that correspond to the sentence (\mathbf{h} itself), its subject, its VP and its direct object. A VP hypothesis has only the nodes $H = \{h_{VP}, h_v, h_{dobj}\}$. In both cases, h_{dobj} may be empty. Both of the denotational similarities $nPMI_{\square}(h, p)$ and $P_{\square}(h|p)$ for $h \in H, p \in P$ lead to two constituent-specific features, sum_x and max_x , (e.g. $\text{sum}_{sbj} = \sum_p \text{sim}(h_{sbj}, p)$, $\text{max}_{dobj} = \max_p \text{sim}(h_{dobj}, p)$) and two global features $\text{sum}_{p,h} = \sum_{p,h} \text{sim}(h, p)$ and $\text{max}_{p,h} = \max_{p,h} \text{sim}(h, p)$. Each constituent type also has a set of node-specific $\text{sum}_{x,s}$ and $\text{max}_{x,s}$ features that are on when constituent x in \mathbf{h} is equal to the string s and whose value is equal to the constituent-based feature. For P_{\square} , each constituent (and each constituent-node pair) has an additional feature $P(h|P) = 1 - \prod_n (1 - P_{\square}(h|p_n))$ that estimates the probability that h is generated by some node in the premise.

Lexical Similarity Features We use two symmetric lexical similarities: standard cosine distance (cos), and Lin (1998)’s similarity (Lin):

$$\begin{aligned} \text{cos}(\mathbf{w}, \mathbf{w}') &= \frac{\mathbf{w} \cdot \mathbf{w}'}{\|\mathbf{w}\| \|\mathbf{w}'\|} \\ \text{Lin}(\mathbf{w}, \mathbf{w}') &= \frac{\sum_{i: \mathbf{w}(i) > 0 \wedge \mathbf{w}'(i) > 0} \mathbf{w}(i) + \mathbf{w}'(i)}{\sum_i \mathbf{w}(i) + \sum_i \mathbf{w}'(i)} \end{aligned}$$

We use two directed lexical similarities: Clarke (2009)’s similarity (Clk), and Szpektor and Dagan (2008)’s balanced precision (Bal), which builds on Lin and on Weeds and Weir (2003)’s similarity (\mathbf{W}):

$$\begin{aligned} \text{Clk}(\mathbf{w} \mid \mathbf{w}') &= \frac{\sum_{i:\mathbf{w}(i)>0 \wedge \mathbf{w}'(i)>0} \min(\mathbf{w}(i), \mathbf{w}'(i))}{\sum_i \mathbf{w}(i)} \\ \text{Bal}(\mathbf{w} \mid \mathbf{w}') &= \sqrt{\mathbf{W}(\mathbf{w} \mid \mathbf{w}') \times \text{Lin}(\mathbf{w}, \mathbf{w}')} \\ \mathbf{W}(\mathbf{w} \mid \mathbf{w}') &= \frac{\sum_{i:\mathbf{w}(i)>0 \wedge \mathbf{w}'(i)>0} \mathbf{w}(i)}{\sum_i \mathbf{w}(i)} \end{aligned}$$

We also use two publicly available resources that provide precomputed similarities, Kotlerman et al. (2010)’s DIRECT noun and verb rules and Chklovski and Pantel (2004)’s VERBOCEAN rules. Both are motivated by the need for numerically quantifiable semantic inferences between predicates. We only use entries that correspond to single tokens (ignoring e.g. phrasal verbs).

Each lexical similarity results in the following features: words in the output are represented by a max-sim_w feature which captures its maximum similarity with any word in the premises ($\text{max-sim}_w = \max_{w' \in P} \text{sim}(w, w')$) and by a sum-sim_w feature which captures the sum of its similarities to the words in the premises ($\text{sum-sim}_w = \sum_{w' \in P} \text{sim}(w, w')$). Global max sim and sum sim features capture the maximal (resp. total) similarity of any word in the hypothesis to the premise.

We compute distributional and compositional similarities (cos, Lin, Bal, Clk, Σ , Π) on our image captions (“cap”), the BNC and Gigaword. For each corpus C , we map each word w that appears at least 10 times in C to a vector \mathbf{w}_C of the non-negative normalized pointwise mutual information scores (Section 2.4) of w and the 1,000 words (excluding stop words) that occur in the most sentences of C . We generally define $P(w)$ (and $P(w, w')$) as the fraction of sentences in C in which w (and w') occur. To allow a direct comparison between distributional and denotational similarities, we first define $P(w)$ (and $P(w, w')$) over individual captions (“cap”), and then, to level the playing field, we redefine $P(w)$ (and $P(w, w')$) as the fraction of images in whose captions w (and w') occur (“img”), and then we use our lexicon to augment captions with all hypernyms (“+hyp”). Finally, we include BNC and Gigaword similarity features (“all”).

	VP task		S task	
Baseline 1: BoW	58.7		71.2	
Baseline 2: BoW-H	59.0		73.6	
External 1: DIRECT	59.2		73.5	
External 2: VerbOcean	60.8		74.0	
	Cap	All	Cap	All
Distributional cos	67.5	71.9	76.1	78.9
Distributional Lin	62.6	70.2	75.4	77.8
Distributional Bal	62.3	69.6	74.7	75.3
Distributional Clk	62.4	69.2	75.4	77.5
Compositional Π	68.4	70.3	75.3	77.3
Compositional Σ	67.8	71.4	76.9	79.2
Compositional Π, Σ	69.8	72.7	77.0	79.6
Denotational $nPMI_{\square}$	74.9		80.2	
Denotational P_{\square}	73.8		79.5	
$nPMI_{\square}, P_{\square}$	75.5		81.2	
Combined cos, Π, Σ	71.1	72.6	77.4	79.2
$nPMI_{\square}, P_{\square}, \Pi, \Sigma$	75.6	75.9	80.2	80.7
$nPMI_{\square}, P_{\square}, \text{cos}$	75.6	75.7	80.2	81.2
$nPMI_{\square}, P_{\square}, \text{cos}, \Pi, \Sigma$	75.8	75.9	81.2	80.5

Table 2: Test accuracy on Approximate Entailment.

Compositional Similarity Features We use two standard compositional baselines to combine the word vectors of a sentence into a single vector: addition ($\mathbf{s}_{\Sigma} = \mathbf{w}_1 + \dots + \mathbf{w}_n$, which can be interpreted as a disjunctive operation), and element-wise (Hadamard) multiplication ($\mathbf{s}_{\Pi} = \mathbf{w}_1 \odot \dots \odot \mathbf{w}_n$, which can be seen as a conjunctive operation). In both cases, we represent the premises (which consist of four captions) as the sum of each caption’s vector $\mathbf{p} = \mathbf{p}_1 + \dots + \mathbf{p}_4$. This gives two compositional similarity features: $\Sigma = \text{cos}(\mathbf{p}_{\Sigma}, \mathbf{h}_{\Sigma})$, and $\Pi = \text{cos}(\mathbf{p}_{\Pi}, \mathbf{h}_{\Pi})$.

6.1 Experimental Results

Table 2 provides the test accuracy of our models on the VP and S tasks. Adding hypernyms (BOW-H) yields a slight improvement over the basic BOW model. Among the external resources, VERBOCEAN is more beneficial than DIRECT, but neither help as much as in-domain distributional similarities (this may be due to sparsity).

Table 2 shows only the simplest (“Cap”) and the most complex (“all”) distributional and compositional models, but Table 3 provides accuracies of these models as we go from standard sentence-based co-occurrence counts towards more denotation graph-like co-occurrence counts that are based on all captions describing the same image (“Img”),

	VP task				S task			
	Cap	Img	+Hyp	All	Cap	Img	+Hyp	All
cos	67.5	69.3	69.8	71.9	76.1	76.8	77.5	78.9
Lin	62.6	63.4	61.3	70.0	75.4	74.8	75.2	77.8
Bal	62.3	61.9	62.8	69.6	74.7	75.5	75.1	75.3
Clk	62.4	67.3	68.0	69.2	75.4	75.5	76.0	77.5
Π	68.4	70.5	70.5	70.3	75.3	76.6	77.1	77.3
Σ	67.8	71.4	71.6	71.4	76.9	78.1	79.1	79.2
Π, Σ	69.8	72.7	72.9	72.7	77.0	78.6	79.3	79.6
$nPMI_{\square}$				74.9				80.2
P_{\square}				73.8				79.5
$nPMI_{\square}, P_{\square}$				75.5				81.2

Table 3: Accuracy on hypotheses as various additions are made to the vector corpora. **Cap** is the image corpus with caption co-occurrence. **Img** is the image corpus with image co-occurrence. **+Hyp** augments the image corpus with hypernyms and uses image co-occurrence. **All** adds the BNC and Gigaword corpora to **+Hyp**.

Words in h	VP task			S task		
	1	2	3+	2	3	4+
% of items	72.8	13.9	13.3	65.3	22.8	11.9
BoW-H	52.0	75.0	80.1	69.1	80.8	84.4
cos (All)	68.8	79.4	81.1	75.9	83.9	85.7
Σ (All)	68.1	80.8	79.5	76.5	83.9	85.1
$nPMI_{\square}$	72.0	82.9	82.2	77.3	85.4	86.2

Table 4: Accuracy on hypotheses of varying length.

include hypernyms (“+Hyp”), and add information from other corpora (“All”). The “+Hyp” column in Table 3 shows that the denotational metrics clearly outperform any distributional metric when both have access to the same information. Although the distributional models benefit from the BNC and Gigaword-based similarities (“All”), their performance is still below that of the denotational models. Among the distributional model, the simple cos performs better than Lin, or the directed Clk and Bal similarities. In all cases, giving models access to different similarity features improves performance.

Table 4 shows the results by hypothesis length. As the length of h increases, classifiers that use similarities between pairs of words (BOW-H and cos) continue to improve in performance relative to the classifiers that use similarities between phrases and sentences (Σ and $nPMI_{\square}$). Most likely, this is due to the lexical similarities having a larger set of features to work with for longer h . $nPMI_{\square}$ does especially well on shorter h , likely due to the shorter h having larger denotations.

7 Task 2: Semantic Textual Similarity

To assess how the denotational similarities perform on a more established task and domain, we apply them to the 1500 sentence pairs from the MSR Video Description Corpus (Chen and Dolan, 2011) that were annotated for the SemEval 2012 Semantic Textual Similarity (STS) task (Agirre et al., 2012). The goal of this task is to assign scores between 0 and 5 to a pair of sentences, where 5 indicates equivalence, and 0 unrelatedness. Since this is a symmetric task, we do not consider directed similarities. And because the goal of this experiment is not to achieve the best possible performance on this task, but to compare the effectiveness of denotational and more established similarities, we only compare the impact of denotational similarities with compositional similarities computed on our own corpus. Since the MSR Video corpus associates each video with multiple sentences, it is in principle also amenable to a denotational treatment, but the STS task description explicitly forbids its use.

7.1 Models

Baseline and Compositional Features Our starting point is Bär et al. (2013)’s DKPro Similarity, one of the top-performing models from the 2012 STS shared task, which is available and easily modified. It consists of a log-linear regression model trained on multiple text features (word and character n -grams, longest common substring and longest common subsequence, Gabrilovich and Markovitch (2007)’s Explicit Semantic Analysis, and Resnik (1995)’s WordNet-based similarity). We investigate the effects of adding compositional (computed on the vectors obtained from the image-caption training data) and denotational similarity features to this state-of-the-art system.

Denotational Features Since the STS task is symmetric, we only consider $nPMI_{\square}$ similarities. We again represent each sentence s by features based on 5 types of constituents: $S = \{s_S, s_{sbj}, s_{VP}, s_v, s_{dobj}\}$. Since sentences might be complex, they might contain multiple constituents of the same type, and we therefore think of each feature as a feature over sets of nodes. For each constituent C we consider two sets of nodes in the denotation graph: C itself (typically leaf nodes),

	<i>DKPro</i>	+ Σ, Π (img)	+ $nPMI_{\square}$	+ <i>both</i>
Pearson r	0.868	0.880	0.888	0.890

Table 5: Performance on the STS MSRvid task: *DKPro* (Bär et al., 2013) plus compositional (Σ, Π) and/or denotational similarities ($nPMI_{\square}$) from our corpus

and C^{anc} , their parents and grandparents. For each pair of sentences, C-C similarities compute the similarity of the constituents of the same type, while C-all similarities compute the similarity of a C constituent in one sentence against all constituents in the other sentence. For each pair of constituents we consider three similarity features: $\text{sim}(C_1, C_2)$, $\max(\text{sim}(C_1 C_2^{anc}), \text{sim}(C_1^{anc}, C_2))$, $\text{sim}(C_1^{anc}, C_2^{anc})$. The similarity of two sets of nodes is determined by the maximal similarity of any pair of their elements: $\text{sim}(C_1, C_2) = \max_{c_1 \in C_1, c_2 \in C_2} nPMI_{\square}(c_1, c_2)$. This gives us 15 C-C features and 15 C-all features.

7.2 Experiments

We use the STS 2012 train/test data, normalized in the same way as the image captions for the denotation graph (i.e. we re-tokenize, lemmatize, and remove determiners). Table 5 shows experimental results for four models: *DKPro* is the off-the-shelf *DKProSimilarity* model (Bär et al., 2013). From our corpus, we either add additive and multiplicative *compositional* features (Σ, Π) from Section 6 (img), the C-C and C-All *denotational* features based on $nPMI_{\square}$, or *both* compositional and denotational features. Systems are evaluated by the Pearson correlation (r) of their predicted similarity scores to the human-annotated ones. We see that the denotational similarities outperform the compositional similarities, and that including compositional similarity features in addition to denotational similarity features has little effect. For additional comparison, the published numbers for the TakeLab Semantic Text Similarity System (Šarić et al., 2012), another top-performing model from the 2012 shared task, are $r = 0.880$ on this dataset.

8 Conclusion

Summary of Contributions We have defined novel *denotational* metrics of linguistic similarity (Section 2), and have shown them to be at least

competitive with, if not superior to, distributional similarities for two tasks that require simple semantic inferences (Sections 6, 7), even though our current method of computing them is somewhat brittle (Section 5). We have also introduced two new resources: a large data set of images paired with descriptive captions, and a *denotation graph* that pairs generalized versions of these captions with their *visual denotations*, i.e. the sets of images they describe. Both of these resources are freely available (<http://nlp.cs.illinois.edu/Denotation.html>) Although the aim of this paper is to show their utility for a purely linguistic task, we believe that they should also be of great interest for people who aim to build systems that automatically associate image with sentences that describe them (Farhadi et al., 2010; Kulkarni et al., 2011; Li et al., 2011; Yang et al., 2011; Mitchell et al., 2012; Kuznetsova et al., 2012; Gupta et al., 2012; Hodosh et al., 2013).

Related Work and Resources We believe that the work reported in this paper has the potential to open up promising new research directions. There are other data sets that pair images or video with descriptive language, but we have not yet applied our approach to them. Chen and Dolan (2011)’s MSR Video Description Corpus (of which the STS data is a subset) is most similar to ours, but its curated part is significantly smaller. Instead of several independent captions, Grubinger et al. (2006)’s IAPR TC-12 data set contains longer descriptions. Ordonez et al. (2011) harvested 1 million images and their user-generated captions from Flickr to create the SBU Captioned Photo Dataset. These captions tend to be less descriptive of the image. The denotation graph is similar to Berant et al. (2012)’s ‘entailment graph’, but differs from it in two ways: first, entailment relations in the denotation graph are defined extensionally in terms of the images described by the expressions at each node, and second, nodes in Berant et al.’s entailment graph correspond to generic propositional templates ($X \text{ treats } Y$), whereas nodes in our denotation graph correspond to complete propositions ($a \text{ dog runs}$).

Acknowledgements

We gratefully acknowledge the support of the National Science Foundation under NSF awards 0803603 “*INT2-Medium: Understanding the meaning of images*”, 1053856 “*CAREER: Bayesian Models for Lexicalized Grammars*”, and 1205627 “*CIP: Collaborative Research: Visual entailment data set and challenge for the Language and Vision Community*”, as well as via an NSF Graduate Research Fellowship to Alice Lai.

References

- Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 task 6: a pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, SemEval '12, pages 385–393.
- Daniel Bär, Torsten Zesch, and Iryna Gurevych. 2013. DKPro Similarity: An Open Source Framework for Text Similarity. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 121–126, Sofia, Bulgaria, August.
- Jon Barwise and John Perry. 1980. Situations and attitudes. *Journal of Philosophy*, 78:668–691.
- Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2012. Learning entailment relations by global graph structure optimization. *Computational Linguistics*, 38(1):73–111.
- Greg Carlson, 2005. *The Encyclopedia of Language and Linguistics*, chapter Generics, Habituals and Iteratives. Elsevier, 2nd edition.
- David Chen and William Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 190–200, Portland, Oregon, USA, June.
- Timothy Chklovski and Patrick Pantel. 2004. Verbocean: Mining the web for fine-grained semantic verb relations. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 33–40, Barcelona, Spain, July.
- Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.
- Daoud Clarke. 2009. Context-theoretic semantics for natural language: an overview. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 112–119, Athens, Greece, March.
- Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL Recognising Textual Entailment challenge. In *Machine Learning Challenges*, volume 3944 of *Lecture Notes in Computer Science*, pages 177–190. Springer.
- David Dowty, Robert Wall, and Stanley Peters. 1981. *Introduction to Montague Semantics*. Reidel, Dordrecht.
- Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every picture tells a story: Generating sentences from images. In *Proceedings of the European Conference on Computer Vision (ECCV), Part IV*, pages 15–29, Heraklion, Greece, September.
- Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th international joint conference on Artificial intelligence*, IJCAI'07, pages 1606–1611.
- Michael Grubinger, Paul Clough, Henning Müller, and Thomas Deselaers. 2006. The IAPR benchmark: A new evaluation resource for visual information systems. In *OntoImage 2006, Workshop on Language Resources for Content-based Image Retrieval during LREC 2006*, pages 13–23, Genoa, Italy, May.
- Ankush Gupta, Yashaswi Verma, and C. Jawahar. 2012. Choosing linguistics over vision to describe images. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, Toronto, Ontario, Canada, July.
- Zellig S Harris. 1954. Distributional structure. *Word*, 10:146–162.
- Micah Hodosh, Peter Young, Cyrus Rashtchian, and Julia Hockenmaier. 2010. Cross-caption coreference resolution for automatic image understanding. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 162–171, Uppsala, Sweden, July.
- Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research (JAIR)*, 47:853–899.
- Shangfeng Hu and Chengfei Liu. 2011. Incorporating coreference resolution into word sense disambiguation. In Alexander F. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 6608 of *Lecture Notes in Computer Science*, pages 265–276. Springer Berlin Heidelberg.

- Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16(4):359–389.
- Girish Kulkarni, Visruth Premraj, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C. Berg, and Tamara L. Berg. 2011. Baby talk: Understanding and generating simple image descriptions. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1601–1608.
- Polina Kuznetsova, Vicente Ordonez, Alexander Berg, Tamara Berg, and Yejin Choi. 2012. Collective generation of natural image descriptions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 359–368, Jeju Island, Korea, July.
- Siming Li, Girish Kulkarni, Tamara L. Berg, Alexander C. Berg, and Yejin Choi. 2011. Composing simple image descriptions using web-scale n-grams. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning (CoNLL)*, pages 220–228, Portland, OR, USA, June.
- Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML)*, pages 296–304, Madison, WI, USA, July.
- Bill MacCartney and Christopher D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 521–528, Manchester, UK, August.
- Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. <http://www.cs.umass.edu/mccallum/mallet>.
- Shachar Mirkin, Ido Dagan, and Eyal Shnarch. 2009. Evaluating the inferential utility of lexical-semantic resources. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 558–566, Athens, Greece, March.
- Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429.
- Margaret Mitchell, Jesse Dodge, Amit Goyal, Kota Yamaguchi, Karl Stratos, Xufeng Han, Alyssa Mensch, Alex Berg, Tamara Berg, and Hal Daume III. 2012. Midge: Generating image descriptions from computer vision detections. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 747–756, Avignon, France, April.
- Richard Montague. 1974. *Formal philosophy: papers of Richard Montague*. Yale University Press, New Haven. Edited by Richmond H. Thomason.
- Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Malt-parser: A data-driven parser-generator for dependency parsing. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 2216–2219.
- Vicente Ordonez, Girish Kulkarni, and Tamara L. Berg. 2011. Im2text: Describing images using 1 million captioned photographs. In *Advances in Neural Information Processing Systems 24*, pages 1143–1151.
- Judita Preiss. 2001. Anaphora resolution with word sense disambiguation. In *Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 143–146, Toulouse, France, July.
- Philip Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 1, IJCAI'95*, pages 448–453.
- Idan Szpektor and Ido Dagan. 2008. Learning entailment rules for unary templates. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 849–856, Manchester, UK, August. Coling 2008 Organizing Committee.
- Frane Šarić, Goran Glavaš, Mladen Karan, Jan Šnajder, and Bojana Dalbelo Bašić. 2012. Takelab: Systems for measuring semantic text similarity. In *Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 441–448, Montréal, Canada, 7-8 June.
- Julie Weeds and David Weir. 2003. A general framework for distributional similarity. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 81–88.
- Yezhou Yang, Ching Teo, Hal Daume III, and Yiannis Aloimonos. 2011. Corpus-guided sentence generation of natural images. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 444–454, Edinburgh, UK, July.

Senti-LSSVM: Sentiment-Oriented Multi-Relation Extraction with Latent Structural SVM

Lizhen Qu
Max Planck Institute
for Informatics
lqu@mpi-inf.mpg.de

Yi Zhang
Nuance Communications
yi.zhang@nuance.com

Rui Wang
DFKI GmbH
mars198356@hotmail.com

Lili Jiang
Max Planck Institute
for Informatics
ljiang@mpi-inf.mpg.de

Rainer Gemulla
Max Planck Institute
for Informatics
rgemulla@mpi-inf.mpg.de

Gerhard Weikum
Max Planck Institute
for Informatics
weikum@mpi-inf.mpg.de

Abstract

Extracting instances of sentiment-oriented relations from user-generated web documents is important for online marketing analysis. Unlike previous work, we formulate this extraction task as a structured prediction problem and design the corresponding inference as an integer linear program. Our latent structural SVM based model can learn from training corpora that do not contain explicit annotations of sentiment-bearing expressions, and it can simultaneously recognize instances of both binary (polarity) and ternary (comparative) relations with regard to entity mentions of interest. The empirical evaluation shows that our approach significantly outperforms state-of-the-art systems across domains (cameras and movies) and across genres (reviews and forum posts). The gold standard corpus that we built will also be a valuable resource for the community.

1 Introduction

Sentiment-oriented relation extraction (Choi et al., 2006) is concerned with recognizing sentiment polarities and comparative relations between entities from natural language text. Identifying such relations often requires syntactic and semantic analysis at both sentence and phrase level. Most prior work on sentiment analysis consider either i) subjective sentence detection (Yu and Kübler, 2011), ii) polarity classification (Johansson and Moschitti, 2011; Wilson et al., 2005), or iii) comparative relation identification (Jindal and Liu, 2006; Ganapathibhotla and Liu, 2008). In practice, however, differ-

ent types of sentiment-oriented relations frequently coexist in documents. In particular, we found that more than 38% of the sentences in our test corpus contain more than one type of relations. The isolated analysis approach is inappropriate because i) it sacrifices accuracy by ignoring the intricate interplay among different types of relations; ii) it could lead to conflicting predictions such as estimating a relation candidate as both negative and comparative. Therefore, in this paper, we identify instances of both sentiment polarities and comparative relations for entities of interest *simultaneously*. We assume that all the mentions of entities and attributes are given, and entities are disambiguated. It is a widely used assumption when evaluating a module in a pipeline system that the outputs of preceding modules are error-free.

To the best of our knowledge, the only existing system capable of extracting both comparisons and sentiment polarities is a rule-based system proposed by Ding et al. (2009). We argue that it is better to tackle the task by using a *unified model* with *structured outputs*. It allows us to consider a set of correlated relation instances *jointly* and characterize their interaction through a set of soft and hard constraints. For example, we can encode constraints to discourage an attribute to participate in a polarity relation and a comparative relation at the same time. As a result, the system extracts a set of correlated instances of sentiment-oriented relations from a given sentence. For example, with the sentence about the camera Canon 7D, “The sensor is great, but the price is higher than Nikon D7000.” the expected output is *positive*(Canon 7D, *sensor*)

and *preferred*(Nikon D7000, Canon 7D, textit-price).

However, constructing a fully annotated training corpus for this task is labor-intensive and requires strong linguistic background. We minimize this overhead by applying a simplified annotation scheme, in which annotators mark mentions of entities and attributes, disambiguate the entities, and label instances of relations for each sentence. Based on the new scheme, we have created a small Sentiment Relation Graph (SRG) corpus for the domains of cameras and movies, which significantly differs from the corpora used in prior work (Wei and Gulla, 2010; Kessler et al., 2010; Toprak et al., 2010; Wiebe et al., 2005; Hu and Liu, 2004) in the following ways: i) both sentiment polarities and comparative relations are annotated; ii) all mentioned entities are disambiguated; and iii) no subjective expressions are annotated, unless they are part of entity mentions.

The new annotation scheme raises a new challenge for learning algorithms in that they need to automatically find textual evidences for each annotated relation during training. For example, with the sentence “*I like the Rebel a little better, but that is another price jump*”, simply assigning a sentiment-bearing expression to the nearest relation candidate is insufficient, especially when the sentiment is not explicitly expressed.

In this paper, we propose SENTI-LSSVM, a latent structural SVM based model for sentiment-oriented relation extraction. SENTI-LSSVM is applied to find the most likely set of the relation instances expressed in a given sentence, where the latent variables are used to assign the most appropriate textual evidences to the respective instances.

In summary, the contributions of this paper are the following:

- We propose SENTI-LSSVM: the first unified statistical model with the capability of extracting instances of both binary and ternary sentiment-oriented relations.
- We design a task-specific integer linear programming (ILP) formulation for inference.
- We construct a new SRG corpus as a valuable asset for the evaluation of sentiment relation

extraction.

- We conduct extensive experiments with on-line reviews and forum posts, showing that SENTI-LSSVM model can effectively learn from a training corpus without explicitly annotated subjective expressions and that its performance significantly outperforms state-of-the-art systems.

2 Related Work

There are ample works on analyzing sentiment polarities and entity comparisons, but the majority of them studied the two tasks in isolation.

Most prior approaches for fine-grained sentiment analysis focus on polarity classification. Supervised approaches on expression-level analysis require the annotation of sentiment-bearing expressions as training data (Jin et al., 2009; Choi and Cardie, 2010; Johansson and Moschitti, 2011; Yessenalina and Cardie, 2011; Wei and Gulla, 2010). However, the corresponding annotation process is time-consuming. Although sentence-level annotations are easier to obtain, the analysis at this level cannot cope with sentences conveying relations of multiple types (McDonald et al., 2007; Täckström and McDonald, 2011; Socher et al., 2012). Lexicon-based approaches require no training data (Ku et al., 2006; Kim and Hovy, 2006; Godbole et al., 2007; Ding et al., 2008; Popescu and Etzioni, 2005; Liu et al., 2005) but suffer from inferior performance (Wilson et al., 2005; Qu et al., 2012). In contrast, our method requires no annotation of sentiment-bearing expressions for training and can predict both sentiment polarities and comparative relations.

Sentiment-oriented comparative relations have been studied in the context of user-generated discourse (Jindal and Liu, 2006; Ganapathibhotla and Liu, 2008). Approaches rely on linguistically motivated rules and assume the existence of independent keywords in sentences which indicate comparative relations. Therefore, these methods fall short of extracting comparative relations based on domain dependent information.

Both Johansson and Moschitti (2011) and Wu et al. (2011) formulate fine-grained sentiment analysis as a learning problem with structured outputs. However, they focus only on polarity classification

of expressions and require annotation of sentiment-bearing expressions for training as well.

While ILP has been previously applied for inference in sentiment analysis (Choi and Cardie, 2009; Somasundaran and Wiebe, 2009; Wu et al., 2011), our task requires a complete ILP reformulation due to 1) the absence of annotated sentiment expressions and 2) the constraints imposed by the joint extraction of both sentiment polarity and comparative relations.

3 System Overview

This section gives an overview of the whole system for extracting sentiment-oriented relation instances. Prior to presenting the system architecture, we introduce the essential concepts and the definitions of two kinds of directed hypergraphs as the representation of correlated relation instances extracted from sentences.

3.1 Concepts and Definitions

Entity. An *entity* is an abstract or concrete thing, which needs not be of material existence. An entity in this paper refers to either a product or a brand.

Attribute. An *attribute* is an object closely associated with or belonging to an entity, such as the lens of digital camera.

Sentiment-Oriented Relation. A *sentiment-oriented relation* is either a sentiment polarity or a comparative relation, defined on tuples of entities and attributes. A *sentiment polarity* relation conveys either a positive or a negative attitude towards entities or their attributes, whereas a *comparative relation* indicates the preference of one entity over the other entity w.r.t. an attribute.

Relation Instance. An instance of *sentiment polarity* takes the form $r(\text{entity}, \text{attribute})$ with $r \in \{\text{positive}, \text{negative}\}$, such as $\text{positive}(\text{Canon 7D}, \text{sensor})$. The polarity instances expressed in the form of unary relations, such as “Nikon D7000 is excellent.”, are denoted as binary relations $r(\text{entity}, \text{whole})$, where the attribute *whole* indicates the entity as a whole. In contrast, an instance of *comparative relation* is in the form of $\text{preferred}\{\text{entity}, \text{entity}, \text{attribute}\}$, e.g. $\text{preferred}(\text{Canon 7D}, \text{Nikon D7000}, \text{price})$. For brevity, we refer to an instance set of sentiment-oriented relations extracted from a

sentence as an **SSOR**. To represent the instances of the remaining relations, we represent them as *other*{entity, attribute}, such as $\text{textitpartOf}\{\text{wheel}, \text{car}\}$. These relations include objective relations and the subjective relations other than sentiment-oriented relations.

Mention-Based Relation Instances. A mention-based relation instance refers to a tuple of entity mentions with a certain relation. This concept is introduced as the representation of instances in a sentence by replacing entities with the corresponding entity mentions, such as $\text{positive}(\text{“Canon SD880i”}, \text{“wide angle view”})$.

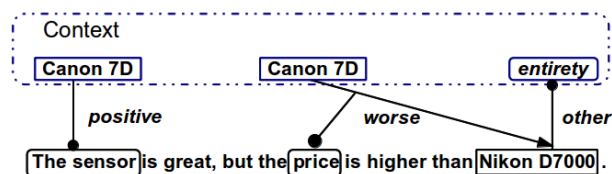


Figure 1: An example of MRG.

Mention-Based Relation Graph. A mention-based relation graph (or MRG) represents a collection of mention-based relation instances expressed in a sentence. As illustrated in Figure 1, an MRG is a directed hypergraph $G = \langle M, E \rangle$ with a vertex set M and an edge set E . A vertex $m_i \in M$ denotes a mention of an entity or an attribute occurring either within the sentence or in its context. We say that a mention is from the context if it is mentioned in the previous sentence or is an attribute implied in the current sentence. An instance of a binary relation in an MRG takes the form of a binary edge $e_l = (m_i, m_a)$, where m_i and m_a denote an entity mention and an attribute mention respectively, and the type $l \in \{\text{positive}, \text{negative}, \text{other}\}$. A ternary edge e_l indicating comparative relation is represented as $e_l = (m_i, m_j, m_a)$, where two entity mentions m_i and m_j are compared with respect to the attribute mention m_a . We define the type $l \in \{\text{better}, \text{worse}\}$ to indicate two possible directions of the relation and assume m_i occurs before m_j . As a result, we have a set L of five relation types: *positive, negative, better, worse* or *other*. According to these definitions, the annotations in the SRG corpus are actually MRGs and disambiguated entities. If there are multiple mentions referring to the same entity, annotators are asked to choose the

most obvious one because it saves annotation time and is less demanding for the entity recognition and diambiguation modules.

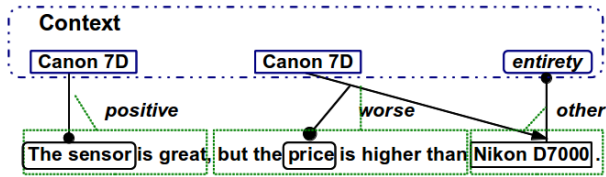


Figure 2: An example of eMRG. The textual evidences are wrapped by green dashed boxes.

Evidentiary Mention-Based Relation Graph. An *evidentiary* mention-based relation graph, coined **eMRG**, extends an MRG by associating each edge with a textual evidence to support the corresponding relation assertions (see Figure 2). Consequently, an edge in an eMRG is denoted by a pair (a, c) , where a represents a mention-based relation instance and c is the associated textual evidence. It is also referred to as an *evidentiary* edge, represented as $e_l = (m_i, m_j, m_a)$, an MRG as an **evidentiary MRG (eMRG)** and the edges of eMRGs as **evidentiary** edges, as shown in Figure 2.

3.2 System Architecture

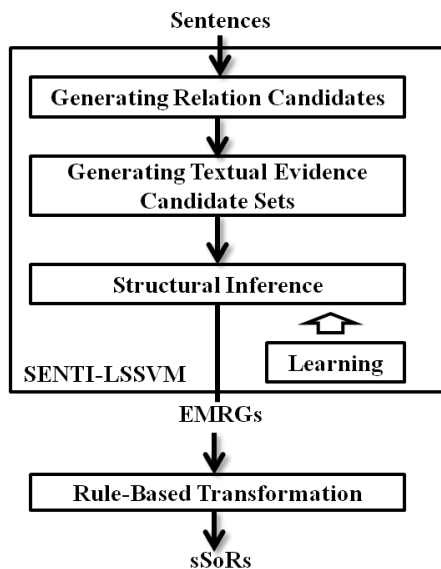


Figure 3: System architecture.

As illustrated by Figure 3, at the core of our system is the **SENTI-LSSVM** model, which extracts sets

of mention-based relationships in the form of eMRGs from sentences. For a given sentence with known entity mentions, we select all possible mention sets as *relation candidates*, where each set includes at least one entity mention. Then we associate each relation candidate with a set of constituents or the whole sentence as the textual evidence candidates (cf. Section 6.1). Subsequently, the inference component aims to find the most likely eMRG from all possible combinations of mention-based relation instances and their textual evidences (cf. Section 6.2). The representation eMRG is chosen because it characterizes exactly the model outputs by letting each edge correspond to an instance of mention-based relation and the associated textual evidence. Finally, the model parameters of this model are learned by an online algorithm (cf. Section 7).

Since instance sets of sentiment-oriented relations (sSoRs) are the expected outputs, we can obtain sSoRs from MRGs by using a simple rule-based algorithm. The algorithm essentially maps the mentions from an MRG into entities and attributes in an sSoR and label the corresponding tuples with the relation types of the edges from an MRG. For instances of comparative relation, the label *better* or *worse* is mapped to the relation type *preferred*.

4 SENTI-LSSVM Model

The task of sentiment-oriented relation extraction is to determine the most likely sSoR in a sentence. Since sSoRs are derived from the corresponding MRGs as described in Section 3, the task is reduced to find the most likely MRG for each sentence. Since an MRG is created by assigning relation types to a subset of all *relation candidates*, which are possible tuples of mentions with unknown relation types, the number of MRGs can be extremely high.

To tackle the task, one solution is to employ an edge-factored linear model in the framework of structural SVM (Martins et al., 2009; Tsochantaridis et al., 2004). The model suggests that a bag of features should be specified for each relation candidate, and then the model predicts the most likely candidate sets along with their relation types to form the optimal MRGs. As we observed, for a relation candidate, the most informative features are the words near its entity mentions in the original text. How-

ever, if we represent a candidate by all these words, it is very likely that the instances of different relation types share overly similar features, because a mention is often involved in more than one relation candidate, as shown in Figure 2. As a consequence, the instances of different relations represented by overly similar features can easily confuse the learning algorithm. Thus, it is critical to select proper constituents or sentences as textual evidences for each relation candidate in both training and testing.

Consequently, we divide the task of sentiment-oriented relation extraction into two subtasks : i) identifying the most likely MRGs; ii) assigning proper textual evidences to each edge of MRGs to support their relation assertions. It is desirable to carry out the two subtasks *jointly* as these two subtasks could enhance each other. First, the identification of relation types requires proper textual evidences; second, the soft and hard constraints imposed by the correlated relation instances facilitate the recognition of the corresponding textual evidences. Since the eMRGs are created by attaching every MRG with a set of textual evidences, tackling the two subtasks simultaneously is equivalent to selecting the most likely eMRG from a set of eMRG candidates. It is challenging because our SRG corpus does not contain any annotation of textual evidences.

Formally, let \mathcal{X} denote the set of all available sentences, and we define $y \in \mathcal{Y}(x) (x \in \mathcal{X})$ as the set of labeled edges of an MRG and $\mathcal{Y} = \cup_{x \in \mathcal{X}} \mathcal{Y}(x)$. Since the assignments of textual evidences are not observed, an assignment of evidences to y is denoted by a latent variable $h \in \mathcal{H}(x)$ and $\mathcal{H} = \cup_{x \in \mathcal{X}} \mathcal{H}(x)$. Then (y, h) corresponds to an eMRG, and $(a, c) \in (y, h)$ is a labeled edge a attached with a textual evidence c . Given a labeled dataset $D = \{(x_1, y_1), \dots, (x_n, y_n)\} \in (\mathcal{X} \times \mathcal{Y})^n$, we aim to learn a discriminant function $f : \mathcal{X} \rightarrow \mathcal{Y} \times \mathcal{H}$ that outputs the optimal eMRG $(y, h) \in \mathcal{Y}(x) \times \mathcal{H}(x)$ for a given sentence x .

Due to the introduction of latent variables, we adopt the latent structural SVM (Yu and Joachims, 2009) for structural classification. Our discriminant function is defined as

$$f(x) = \operatorname{argmax}_{(y,h) \in \mathcal{Y}(x) \times \mathcal{H}(x)} \beta^\top \Phi(x, y, h) \quad (1)$$

where $\Phi(x, y, h)$ is the feature function of an eMRG (y, h) and β is the corresponding weight vector.

To ensure tractability, we also employ edge-based factorization for our model. Let M_p denote a set of entity mentions and $y_r(m_i)$ be a set of edges labeled with sentiment-oriented relations incident to m_i , the factorization of $\Phi(x, y, h)$ is given as

$$\Phi(x, y, h) = \sum_{(a,c) \in (y,h)} \Phi_e(x, a, c) + \sum_{m_i \in M_p} \sum_{a, a' \in y_r(m_i), a \neq a'} \Phi_c(a, a') \quad (2)$$

where $\Phi_e(x, a, c)$ is a local edge feature function for a labeled edge a attached with a textual evidence c and $\Phi_c(a, a')$ is a feature function capturing co-occurrence of two labeled edges a_{m_i} and a'_{m_i} incident to an entity mention m_i .

5 Feature Space

The following features are used in the feature functions (Equation 2):

Unigrams: As mentioned before, a textual evidence attached to an edge in MRG is either a word, phrase or sentence. We consider all lemmatized unigrams in the textual evidence as unigram features.

Context: Since web users usually express related sentiments about the same entity across sentence boundaries, we describe the sentiment flow using a set of contextual binary features. For example, if entity A is mentioned in both the previous sentence and the current sentence, a set of contextual binary features are used to indicate all possible combinations of the current and the previous mentioned sentiment-oriented relations regarding to entity A.

Co-occurrence: We have mentioned the co-occurrence feature in Equation 2, indicated by $\Phi_c(a, a')$. It captures the co-occurrence of two labeled edges incident to the same entity mention. Note that the co-occurrence feature function is considered only if there is a contrast conjunction such as “*but*” between the non-shared entity mentions incident to the two labeled edges.

Senti-predictors: Following the idea of (Qu et al., 2012), we encode the prediction results from the rule-based phrase-level multi-relation predictor (Ding et al., 2009) and from the bag-of-opinions predictor (Qu et al., 2010) as features based on the textual evidence. The output of the first predictor is an integer value, while the output of the second predictor is a sentiment relation, such as “positive”,

“negative”, “better” or “worse”. We map the relational outputs into integer values and then encode the outputs from both predictors as senti-predictor features.

Others: The commonly used part-of-speech tags are also included as features. Moreover, for an edge candidate, a set of binary features are used to denote the types of the edge and its entity mentions. For instance, a binary feature indicates whether an edge is a binary edge related to an entity mentioned in context. To characterize the syntactic dependencies between two adjacent entity mentions, we use the path in the dependency tree between the heads of the corresponding constituents, the number of words and other mentions in-between as features. Additionally, if the textual evidence is a constituent, its feature w.r.t. an edge is the dependency path to the closest mention of the edge that does not overlap with this constituent.

6 Structural Inference

In order to find the best eMRG for a given sentence with a well trained model, we need to determine the most likely relation type for each relation candidate and support the corresponding assertions with proper textual evidences. We formulate this task as an Integer Linear Programming (ILP). Instead of considering all constituents of a sentence, we empirically select a subset as textual evidences for each relation candidate.

6.1 Textual Evidence Candidates Selection

Textual evidences are selected based on the constituent trees of sentences parsed by the Stanford parser (Klein and Manning, 2003). For each mention in a sentence, we first locate a constituent in the tree with the maximal overlap by Jaccard similarity. Starting from this constituent, we consider two types of candidates: *type I* candidates are constituents at the highest level which contain neither any word of another mention nor any contrast conjunctions such as “*but*”; *type II* candidates are constituents at the highest level which cover exactly two mentions of an edge and do not overlap with any other mentions. For a binary edge connecting an entity mention and an attribute mention, we consider a *type I* candidate starting from the attribute men-

tion. For a binary edge connecting two entity mentions, we consider *type I* candidates starting from both mentions. Moreover, for a comparative ternary edge, we consider both *type I* and *type II* candidates starting from the attribute mention. This strategy is based on our observation that these candidates often cover the most important information w.r.t. the covered entity mentions.

6.2 ILP Formulation

We formulate the inference problem of finding the best eMRG as an ILP problem due to its convenient integration of both soft and hard constraints.

Given the model parameters β , we reformulate the score of an eMRG in the discriminant function (1) as follows,

$$\beta^\top \Phi(x, y, h) = \sum_{(a,c) \in (y,h)} s_{ac} z_{ac} + \sum_{m_i \in M_p} \sum_{a,a' \in y_r(m_i), a \neq a'} s_{aa'} z_{aa'}$$

where $s_{ac} = \beta^\top \Phi_e(x, a, c)$ denotes the score of a labeled edge a attached with a textual evidence c , $s_{aa'} = \beta^\top \Phi_c(a, a')$ is the edge co-occurrence score, the binary variable z_{ac} indicates the presence or absence of the corresponding edge, and $z_{aa'}$ indicates if two edges co-occur. As not every edge set can form an eMRG, we require that a valid eMRG should satisfy a set of linear constraints, which form our *constraint space*. Then function (1) is equivalent to

$$\begin{aligned} \max_{\mathbf{z} \in \mathbb{B}} \quad & \mathbf{s}^\top \mathbf{z} + \mu \mathbf{z}^d \\ \text{s.t.} \quad & \mathbf{A} \begin{bmatrix} \mathbf{z} \\ \boldsymbol{\eta} \\ \boldsymbol{\tau} \end{bmatrix} \leq \mathbf{d} \\ & \mathbf{z}, \boldsymbol{\eta}, \boldsymbol{\tau} \in \mathbb{B} \end{aligned}$$

where $\mathbb{B} = 2^S$ with $S = \{0, 1\}$, and $\boldsymbol{\eta}$ and $\boldsymbol{\tau}$ are auxiliary binary variables that help define the constraint space. The above optimization problem takes exactly the form of an ILP because both the constraints and the objective function are linear, and all variables take only integer values.

In the following, we consider two types of constraint space, 1) an eMRG with only binary edges and 2) an eMRG with both binary and ternary edges.

eMRG with only Binary Edges: An eMRG has only binary edges if a sentence contains no attribute mention or at most one entity mention. We expect that each edge has only one relation type and is supported by a single textual evidence. To facilitate the formulation of constraints, we introduce η_{e_l} to denote the presence or absence of a labeled edge e_l , and η_{ec} to indicate if a textual evidence c is assigned to an unlabeled edge e . Then the binary variable for the corresponding evidentiary edge $z_{e_l c} = \eta_{ec} \wedge \eta_{e_l}$, where the ILP formulation of conjunction can be found in (Martins et al., 2009).

Let C_e denote the set of textual evidence candidates of an unlabeled edge e . The constraint of *at most one textual evidence per edge* is formulated as:

$$\sum_{c \in C_e} \eta_{ec} \leq 1 \quad (3)$$

Once a textual evidence is assigned to an edge, their relation labels should match and the number of labeled edges must agree with the number of attached textual evidences. Further, we assume that a textual evidence c conveys at most one relation so that an evidence will not be assigned to the relations of different types, which is the main problem for the structural SVM based model. Let η_{cl} indicate that the textual evidence c is labeled by the relation type l . The corresponding constraints are expressed as,

$$\sum_{l \in L_e} \eta_{e_l} = \sum_{c \in C_e} \eta_{ec}; \quad z_{e_l c} \leq \eta_{cl}; \quad \sum_{l \in L} \eta_{cl} \leq 1$$

where L_e denotes the set of all possible labels for an unlabeled edge e , and L is the set of all relation types of MRGs (cf. Section 3).

In order to avoid a textual evidence being overly reused by multiple relation candidates, we first penalize the assignment of a textual evidence c to a labeled edge a by associating the corresponding z_{ac} with a fixed negative cost $-\mu$ in the objective function. Then the selection of one textual evidence per edge a is encouraged by associating μ to z_c^d in the objective function, where $z_c^d = \bigvee_{e \in S_c} \eta_{ec}$ and S_c is the set of edges that the textual evidence c serves as a candidate. The disjunction z_c^d is expressed as:

$$\begin{aligned} z_c^d &\geq \eta_e, e \in S_c \\ z_c^d &\leq \sum_{e \in S_c} \eta_e \end{aligned}$$

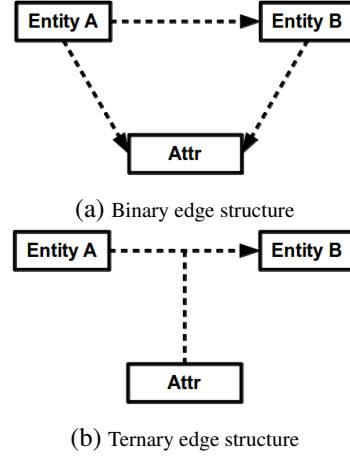


Figure 4: Alternative structures associated with an attribute mention.

This soft constraint not only encourages one textual evidence per edge, but also keeps it eligible for multiple assignments.

For any two labeled edge a and a' incident to the same entity mention, the edge-to-edge co-occurrence is described by $z_{a,a'}^c = z_a \wedge z_{a'}$.

eMRG with both Binary and Ternary Edges: If there are more than one entity mentions and at least one attribute mention in a sentence, an eMRG can potentially have both binary and ternary edges. In this case, we assume that each mention of attributes can participate either in binary relations or in ternary relations. The assumption holds in more than 99.9% of the sentences in our SRG corpus, thus we describe it as a set of hard constraints. Geometrically, the assumption can be visualized as the selection between two alternative structures incident to the same attribute mention, as shown in Figure 4. Note that, in the binary edge structure, we include not only the edges incident to the attribute mention but also the edge between the two entity mentions.

Let $S_{m_i}^b$ be the set of all possible labeled edges in a binary edge structure of an attribute mention m_i . Variable $\tau_{m_i}^b = \bigvee_{e_l \in S_{m_i}^b} \eta_{e_l}$ indicates whether the attribute mention is associated with a binary edge structure or not. In the same manner, we use $\tau_{m_i}^t = \bigvee_{e_l \in S_{m_i}^t} \eta_{e_l}$ to indicate the association of the an attribute mention m_i with an ternary edge structure from the set of all incident ternary edges $S_{m_i}^t$. The selection between two alternative structures is

formulated as $\tau_{m_i}^b + \tau_{m_i}^t = 1$. As this influences only the edges incident to an attribute mention, we keep all the constraints introduced in the previous section unchanged except for constraint (3), which is modified as

$$\sum_{c \in C_e} \eta_{ec} \leq \tau_{m_i}^b; \quad \sum_{c \in C_e} \eta_{ec} \leq \tau_{m_i}^t$$

Therefore, we can have either binary edges or ternary edges for an attribute mention.

7 Learning Model Parameters

Given a set of training sentences $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$, the best weight vector β of the discriminant function (1) is found by solving the following optimization problem:

$$\min_{\beta} \frac{1}{n} \sum_{i=1}^n \left[\max_{(\hat{y}, \hat{h}) \in \mathcal{Y}(x) \times \mathcal{H}(x)} (\beta^\top \Phi(x, \hat{y}, \hat{h}) + \delta(\hat{h}, \hat{y}, y)) - \max_{\bar{h} \in \mathcal{H}(x)} \beta^\top \Phi(x, y, \bar{h}) \right] + \rho \|\beta\| \quad (4)$$

where $\delta(\hat{h}, \hat{y}, y)$ is a loss function measuring the discrepancies between an eMRG (y, \bar{h}) with gold standard edge labels y and an eMRG (\hat{y}, \hat{h}) with inferred labeled edges \hat{y} and textual evidences \hat{h} . Due to the sparse nature of the lexical features, we apply L1 regularizer to the weight vector β , and the degree of sparsity is controlled by the hyperparameter ρ .

Since the L1 norm in the above optimization problem is not differentiable at zero, we apply the online *forward-backward splitting* (FOBOS) algorithm (Duchi and Singer, 2009). It requires two steps for updating the weight vector β by using a single training sentence x on each iteration t .

$$\begin{aligned} \beta_{t+\frac{1}{2}} &= \beta_t - \varepsilon_t \Delta_t \\ \beta_{t+1} &= \arg \min_{\beta} \frac{1}{2} \|\beta - \beta_{t+\frac{1}{2}}\|^2 + \varepsilon_t \rho \|\beta\| \end{aligned}$$

where Δ_t is the subgradient computed without considering the L1 norm and ε_t is the learning rate. For a labeled sentence x , $\Delta_t = \Phi(x, \hat{y}^*, \hat{h}^*) - \Phi(x, y, \bar{h}^*)$, where the feature functions of the corresponding eMRGs are inferred by solving $(\hat{y}^*, \hat{h}^*) = \arg \max_{(\hat{y}, \hat{h}) \in \mathcal{Y}(x) \times \mathcal{H}(x)} [\beta^\top \Phi(x, \hat{y}, \hat{h}) + \delta(\hat{h}, \hat{y}, y)]$ and $(y, \bar{h}^*) = \arg \max_{\bar{h} \in \mathcal{H}(x)} \beta^\top \Phi(x, y, \bar{h})$, as indicated in the optimization problem (4).

The former inference problem is similar to the one we considered in the previous section except the inclusion of the loss function. We incorporate the loss function into the ILP formulation by defining the loss between an MRG (y, h) and a gold standard MRG as the sum of per-edge costs. In our experiments, we consider a positive cost φ for each wrongly labeled edge a , so that if an edge a has a different label from the gold standard, we add φ to the coefficient s_{ac} of the corresponding variable z_{ac} in the objective function of the ILP formulation.

In addition, since the non-positive weights of edge labels in the initial learning phrase often lead to eMRGs with many unlabeled edges, which harms the learning performance, we fix it by adding a constraint for the minimal number of labeled edges in an eMRG,

$$\sum_{a \in A} \sum_{c \in C_a} \eta_{ac} \geq \zeta \quad (5)$$

where A is the set of all labeled edge candidates and ζ denotes the minimal number of labeled edges.

Empirically, the best way to determine ζ is to make it equal to the maximal number of labeled edges in an eMRG with the restriction that a textual evidence can be assigned to at most one edge. By considering all the edge candidates A and all the textual evidence candidates C as two vertex sets in a bipartite graph $\hat{G} = \langle V = (A, C), E \rangle$ (with edges in E indicating which textual evidence can be assigned to which edge), ζ corresponds to exactly the size of a maximum matching of the bipartite graph¹.

To find the optimal eMRG (y, \bar{h}^*) , for the gold label k of each edge, we consider the following set of constraints for inference since the labels of the edges are known for the training data,

$$\begin{aligned} \sum_{c \in C_e} \eta_{ec} &\leq 1; & \eta_{ec} &\leq l_{ck} \\ \sum_{k' \in L} l_{ck'} &\leq 1; & \sum_{e \in S_c} \eta_{ec} &\leq 1 \end{aligned}$$

We include also the soft constraints, which avoid a textual evidence being overly reused by multiple relations, and the constraints similar to (5) to ensure a minimal number of labeled edges and a minimal number of sentiment-oriented relations.

¹It is computed by the Hopcroft-Karp algorithm (Hopcroft and Karp, 1973) in our implementation.

8 SRG Corpus

For evaluation we constructed the SRG corpus, which in total consists of 1686 manually annotated online reviews and forum posts in the digital camera and movie domains². For each domain, we maintain a set of attributes and a list of entity names.

The annotation scheme for the sentiment representation asserts minimal linguistic knowledge from our annotators. By focusing on the meanings of the sentences, the annotators make decisions based on their language intuition, not restricted by specific syntactic structures. Taking the example in Figure 2, the annotators only need to mark the mentions of entities and attributes from both the sentences and the context, disambiguate them, and label (“Canon 7D”, “Nikon D7000”, *price*) as *worse* and (“Canon 7D”, “sensor”) as *positive*, whereas in prior work, people have annotated the sentiment-bearing expressions such as “great” and link them to the respective relation instances as well. This also enables them to annotate instances of both sentiment polarity and comparative relation, which are conveyed by not only explicit sentiment-bearing expressions like “*excellent performance*”, but also factual expressions implying evaluations such as “*The 7V has 10x optical zoom and the 9V has 16x.*”.

	Camera		Movie	
	Reviews	Forums	Reviews	Forums
<i>positive</i>	386	1539	879	905
<i>negative</i>	165	363	529	331
<i>comparison</i>	30	480	39	35

Table 1: Distribution of relation instances in SRG corpus.

14 annotators participated in the annotation project. After a short training period, annotators worked on randomly assigned documents one at a time. For product reviews, the system lists all relevant information about the entity and the predefined attributes. For forum posts, the system shows only the attribute list. For each sentence in a document, the annotator first determines if it refers to an entity of interest. If not, the sentence is marked

²The 107 camera reviews are from *bestbuy.com* and *Amazon.com*; the 667 camera forum posts are downloaded from *forum.digitalcamerareview.com*; the 138 movie reviews and 774 forum posts are from *imdb.com* and *boards.ie* respectively

as *off-topic*. Otherwise, the annotator will identify the most obvious mentions, disambiguate them, and mark the MRGs. We evaluate the inter-annotator agreement on sSoRs in terms of Cohen’s Kappa (κ) (Cohen, 1968). An average Kappa value of 0.698 was achieved on a randomly selected set consisting of 412 sentences.

Table 1 shows the corpus distribution after normalizing them into sSoRs. Camera forum posts contain the largest proportion of comparisons because they are mainly about the recommendation of digital cameras. In contrast, web users are much less interested in comparing movies, in both reviews and forums. In all subsets, positive relations play a dominant role since web users intend to express more positive attitudes online than negative ones (Pang and Lee, 2007).

9 Experiments

This section describes the empirical evaluation of SENTI-LSSVM together with two competitive baselines on the SRG corpus.

9.1 Experimental Setup

We implemented a rule-based baseline (DING-RULE) and a structural SVM (Tsochantaridis et al., 2004) baseline (SENTI-SSVM) for comparison. The former system extends the work of Ding et al. (2009), which designed several linguistically-motivated rules based on a sentiment polarity lexicon for relation identification and assumes there is only one type of sentiment relation in a sentence. In our implementation, we keep all the rules of (Ding et al., 2009) and add one phrase-level rule when there are more than one mention in a sentence. The additional rule assigns sentiment-bearing words and negators to its nearest relation candidates based on the absolute surface distance between the words and the corresponding mentions. In this case, the phrase-level sentiment-oriented relations depend only on the assigned sentiment words and negators. The latter system is based on a structural SVM and does not consider the assignment of textual evidences to relation instances during inference. The textual features of a relation candidate are all lexical and sentiment predictor features within a surface distance of four words from the mentions of the candidate.

Thus, this baseline does not need the inference constraints of SENTI-LSSVM for the selection of textual evidences. To gain more insights into the model, we also evaluate the contribution of individual features of SENTI-LSSVM. In addition, to show if identifying sentiment polarities and comparative relations *jointly* works better than tackling each task on its own, we train SENTI-LSSVM for each task separately and combine their predictions according to compatibility rules and the corresponding graph scores.

For each domain and text genre, we withheld 15% documents for development and use the remaining for cross validation. The hyperparameters of all systems are tuned on the development datasets. For all experiments of SENTI-LSSVM, we use $\rho = 0.0001$ for the L1 regularizer in Eq.(4) and $\varphi = 0.05$ for the loss function; and for SENTI-SSVM, $\rho = 0.0001$ and $\varphi = 0.01$. Since the relation type of *off-topic* sentences is certainly *other*, we evaluate all systems with 5-fold cross-validation only on the *on-topic* sentences in the evaluation dataset. Since the same sSoR can have several equivalent MRGs and the relation type *other* is not of our interest, we evaluate the sSoRs in terms of precision, recall and F-measure. All reported numbers are averages over the 5 folds.

9.2 Results

Table 2 shows the complete results of all systems. Here our model SENTI-LSSVM outperformed all baselines in terms of the average F-measure scores and recalls by a large margin. The F-measure on movie reviews is about 14% over the best baseline. The rule-based system has higher precision than recall in most cases. However, simply increasing the coverage of the domain independent sentiment polarity lexicon might lead to worse performance (Taboada et al., 2011) because many sentiment oriented relations are conveyed by domain dependent expressions and factual expressions implying evaluations, such as “*This camera does not have manual control.*” Compared to DING-RULE, SENTI-SSVM performs better in the camera domain but worse for the movies due to many misclassification of *negative* relation instances as *other*. It also wrongly predicted more *positive* instances as *other* than SENTI-LSSVM. We found that the recalls of these instances are low because they often have overly similar features with the instances of the type

other linking to the same mentions. The problem gets worse in the movie domain since i) many sentences contain no explicit sentiment-bearing words; ii) the prior polarity of the sentiment-bearing words do not agree with their contextual polarity in the sentences. Consider the following example from a forum post about the movie “*Superman Returns*”: “*Have a look at Superman: the Animated Series or Justice League Unlimited ... that is how the characters of Superman and Lex Luthor should be.*”. In contrast, our model minimizes the overlapping features by assigning them to the most likely relation candidates. This leads to significantly better performance. Although SENTI-SSVM has low recall for both *positive* and *negative* relations, it achieves the highest recall for the comparative relation among all systems in the movie domain and camera reviews. Since less than 1% of all instances are for comparative relations in these document sets and all models are trained to optimize the overall accuracy, SENTI-LSSVM intends to trade off the minority class for the overall better performance. This advantage disappears on the camera forum posts, where the number of instances of comparative relation is 12 times more than that in the other data sets.

All systems perform better in predicting positive relations than the negative ones. This corresponds well to the empirical findings in (Wilson, 2008) that people intend to use more complex expressions for negative sentiments than their affirmative counterparts. It is also in accordance with the distribution of these relations in our SRG corpus which is randomly sampled from the online documents. For learning systems, it can also be explained by the fact that the training data for positive relations are considerably more than those for negative ones. The comparative relation is the hardest one to process since we found that many corresponding expressions do not contain explicit keywords for comparison.

To understand the performance of the key feature groups in our model better, we remove each group from the full SENTI-LSSVM system and evaluate the variations with movie reviews and camera forum posts, which have relatively balanced distribution of relation types. As shown in Table 3, the features from the sentiment predictors make significant contributions for both datasets. The different drops of the performance indicate that the po-

		Positive			Negative			Comparison			Micro-average		
		P	R	F	P	R	F	P	R	F	P	R	F
Camera Forum	DING-RULE	56.4	39.0	46.1	46.2	24.0	31.6	42.6	14.0	21.0	53.4	30.8	39.0
	SENTI-SSVM	60.2	35.6	44.8	44.2	38.5	41.2	28.0	40.1	32.9	43.7	36.7	39.9
	SENTI-LSSVM	69.2	38.9	49.8	50.8	39.3	44.3	42.6	35.1	38.5	56.5	38.0	45.4
Camera Re-view	DING-RULE	83.6	69.0	75.6	68.6	38.8	49.6	30.0	16.9	21.6	81.1	58.6	68.1
	SENTI-SSVM	72.6	75.4	74.0	63.9	62.5	63.2	28.0	38.9	32.5	68.1	70.4	69.3
	SENTI-LSSVM	77.3	85.4	81.2	68.9	61.3	64.9	22.3	20.7	21.6	73.1	73.4	73.7
Movie Forum	DING-RULE	63.7	37.4	47.1	27.6	34.3	30.6	8.9	5.6	6.8	48.2	35.9	41.2
	SENTI-SSVM	66.2	30.1	41.3	25.6	17.3	20.7	44.2	56.7	49.7	53.3	27.9	36.6
	SENTI-LSSVM	63.3	44.2	52.1	29.7	45.6	36.0	40.1	45.0	42.4	49.7	44.6	47.0
Movie Re-view	DING-RULE	66.5	47.2	55.2	42.0	39.1	40.5	31.4	12.0	17.4	56.2	44.0	49.4
	SENTI-SSVM	61.3	54.0	57.4	45.2	13.7	21.1	24.5	63.3	35.3	54.6	39.2	45.7
	SENTI-LSSVM	59.0	79.1	67.6	53.3	51.4	52.3	28.3	34.0	30.9	57.9	68.8	62.9

Table 2: Evaluation results for DING-RULE, SENTI-SSVM and SENTI-LSSVM. Boldface figures are statistically significantly better than all others in the same comparison group under t-test with $p = 0.05$.

Feature Models	Movie Reviews	Camera Forums
full system	62.9	45.4
¬unigram	63.2 (+0.3)	41.2 (-4.2)
¬context	54.5 (-8.4)	46.0 (+0.6)
¬co-occurrence	62.6 (-0.3)	44.9 (-0.5)
¬senti-predictors	61.3 (-1.6)	34.3 (-11.1)

Table 3: Micro-average F-measure of SENTI-LSSVM with different feature models

larities predicted by rules are more consistent in camera forum posts than in movie reviews. Due to the complexity of expressions in the movie reviews our model cannot benefit from the unigram features but these features are a good compensation for the sentiment predictor features in camera forum posts. The sharp drop by removing the context features from our model on movie reviews indicates that the sentiments in movie reviews depend highly on the relations of the previous sentences. In contrast, the sentiment-oriented relations of the previous sentences could be a reason of overfitting for camera forum data. The edge co-occurrence features do not play an important role in our model since the number of co-occurred sentiment-oriented relations in the sentences with contrast conjunctions like “*but*” is small. However, we found that allowing the co-occurrence of any sentiment-oriented relations would harm the performance of the model.

In addition, our experiments showed that the sep-

arated approach, which trains a model for sentiment polarities and comparative relations respectively, leads to a decrease by almost 1% in terms of the F-measure averaged over all four datasets. The largest drop of F-measure is 3% on camera forum posts, since this dataset contains the largest proportion of comparative relations. We found that the errors are increased when the trained models make conflicting predictions. In this case, the joint approach can take all factors into account and make more consistent decisions than the separated approaches.

10 Conclusion

We proposed SENTI-LSSVM model for extracting instances of both sentiment polarities and comparative relations. For evaluating and training the model, we created an SRG corpus by using a lightweight annotation scheme. We showed that our model can automatically find textual evidences to support its relation predictions and achieves significantly better F-measure scores than alternative state-of-the-art methods.

References

Yejin Choi and Claire Cardie. 2009. Adapting a polarity lexicon using integer linear programming for domain-specific sentiment classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural*

- Language Processing: Volume 2 - Volume 2*, EMNLP '09, pages 590–598, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Yejin Choi and Claire Cardie. 2010. Hierarchical sequential learning for extracting opinions and their attributes. In *Proceedings of the Annual meeting of the Association for Computational Linguistics*, pages 269–274. Association for Computational Linguistics.
- Yejin Choi, Eric Breck, and Claire Cardie. 2006. Joint extraction of entities and relations for opinion recognition. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 431–439, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Jacob Cohen. 1968. Weighted Kappa: Nominal Scale Agreement Provision for Scaled Disagreement or Partial Credit. *Psychological bulletin*, 70(4):213.
- Xiaowen Ding, Bing Liu, and Philip S. Yu. 2008. A holistic lexicon-based approach to opinion mining. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 231–240, New York, NY, USA. ACM.
- Xiaowen Ding, Bing Liu, and Lei Zhang. 2009. Entity discovery and assignment for opinion mining applications. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1125–1134.
- John Duchi and Yoram Singer. 2009. Efficient online and batch learning using forward backward splitting. *The Journal of Machine Learning Research*, 10:2899–2934.
- Murthy Ganapathibhotla and Bing Liu. 2008. Mining opinions in comparative sentences. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, pages 241–248, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Namrata Godbole, Manjunath Srinivasaiah, and Steven Skiena. 2007. Large-scale sentiment analysis for news and blogs (system demonstration). In *Proceedings of the International AAAI Conference on Weblogs and Social Media*.
- John E Hopcroft and Richard M Karp. 1973. An n^2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231.
- Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pages 168–177, New York, NY, USA. ACM.
- Wei Jin, Hung Hay Ho, and Rohini K. Srihari. 2009. Opinionminer: a novel machine learning system for web opinion mining and extraction. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1195–1204, New York, NY, USA. ACM.
- Nitin Jindal and Bing Liu. 2006. Mining comparative sentences and relations. In *Proceedings of the 21st International Conference on Artificial Intelligence - Volume 2*, AAAI'06, pages 1331–1336. AAAI Press.
- Richard Johansson and Alessandro Moschitti. 2011. Extracting opinion expressions and their polarities—exploration of pipelines and joint models. In *Proceedings of the Annual meeting of the Association for Computational Linguistics*, volume 11, pages 101–106.
- Jason S. Kessler, Miriam Eckert, Lyndsie Clark, and Nicolas Nicolov. 2010. The 2010 icwsm jdpa sentiment corpus for the automotive domain. In *4th International AAAI Conference on Weblogs and Social Media Data Workshop Challenge (ICWSM-DWC 2010)*.
- Soo-Min Kim and Eduard Hovy. 2006. Extracting opinions, opinion holders, and topics expressed in online news media text. In *Proceedings of the Workshop on Sentiment and Subjectivity in Text*, SST '06, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Lun-Wei Ku, Yu-Ting Liang, and Hsin-Hsi Chen. 2006. Opinion extraction, summarization and tracking in news and blog corpora. In *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, pages 100–107.
- Bing Liu, Minqing Hu, and Junsheng Cheng. 2005. Opinion observer: analyzing and comparing opinions on the web. In *Proceedings of the 14th international conference on World Wide Web*, pages 342–351, New York, NY, USA. ACM.
- André L. Martins, Noah A. Smith, and Eric P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Annual meeting of the Association for Computational Linguistics*, pages 342–350.
- Ryan T. McDonald, Kerry Hannan, Tyler Neylon, Mike Wells, and Jeffrey C. Reynar. 2007. Structured models for fine-to-coarse sentiment analysis. In *Proceedings of the Annual meeting of the Association for Computational Linguistics*.
- Bo Pang and Lillian Lee. 2007. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

- Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 339–346, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Lizhen Qu, Georgiana Ifrim, and Gerhard Weikum. 2010. The bag-of-opinions method for review rating prediction from sparse text patterns. In Chu-Ren Huang and Dan Jurafsky, editors, *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, ACL Anthology, pages 913–921, Beijing, China. Tsinghua University Press.
- Lizhen Qu, Rainer Gemulla, and Gerhard Weikum. 2012. A weakly supervised model for sentence-level semantic orientation analysis with multiple experts. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 149–159, Jeju Island, Korea, July. Proceedings of the Annual meeting of the Association for Computational Linguistics.
- Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1201–1211.
- Swapna Somasundaran and Janyce Wiebe. 2009. Recognizing stances in online debates. In *Proceedings of the Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 226–234.
- Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberley D. Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational Linguistics*, 37(2):267–307.
- Oscar Täckström and Ryan McDonald. 2011. Discovering fine-grained sentiment with latent variable structured prediction models. In *Proceedings of the 33rd European conference on Advances in information retrieval, ECIR'11*, pages 368–374, Berlin, Heidelberg. Springer-Verlag.
- Cigdem Toprak, Niklas Jakob, and Iryna Gurevych. 2010. Sentence and expression level annotation of opinions in user-generated discourse. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 575–584, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the International Conference on Machine Learning*, pages 104–112.
- Wei Wei and Jon Atle Gulla. 2010. Sentiment learning on product reviews via sentiment ontology tree. In *Proceedings of the Annual meeting of the Association for Computational Linguistics*, pages 404–413.
- Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2-3):165–210.
- Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 347–354, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Theresa Ann Wilson. 2008. *Fine-grained subjectivity and sentiment analysis: recognizing the intensity, polarity, and attitudes of private states*. Ph.D. thesis, UNIVERSITY OF PITTSBURGH.
- Yuanbin Wu, Qi Zhang, Xuanjing Huang, and Lide Wu. 2011. Structural opinion mining for graph-based sentiment representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1332–1341.
- Ainur Yessenalina and Claire Cardie. 2011. Compositional matrix-space models for sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 172–182.
- Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural svms with latent variables. In *Proceedings of the International Conference on Machine Learning*, page 147.
- Ning Yu and Sandra Kübler. 2011. Filling the gap: Semi-supervised learning for opinion detection across domains. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 200–209. Association for Computational Linguistics.

