Semi-Supervised and Latent-Variable Models of Natural Language Semantics

Dipanjan Das

CMU-LTI-12-007

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:

Noah A. Smith, Carnegie Mellon University
William W. Cohen, Carnegie Mellon University
Lori S. Levin, Carnegie Mellon University
Dan Roth, University of Illinois, Urbana-Champaign

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
In Language and Information Technologies*

© 2012, Dipanjan Das

Abstract

This thesis focuses on robust analysis of natural language semantics. A primary bottleneck for semantic processing of text lies in the scarcity of high-quality and large amounts of annotated data that provide complete information about the semantic structure of natural language expressions. In this dissertation, we study statistical models tailored to solve problems in computational semantics, with a focus on modeling structure that is not visible in annotated text data.

We first investigate *supervised* methods for modeling two kinds of semantic phenomena in language. First, we focus on the problem of **paraphrase identification**, which attempts to recognize whether two sentences convey the same meaning. Second, we concentrate on **shallow semantic parsing**, adopting the theory of frame semantics (Fillmore, 1982). Frame semantics offers deep linguistic analysis that exploits the use of lexical semantic properties and relationships among semantic frames and roles. Unfortunately, the datasets used to train our paraphrase and frame-semantic parsing models are too small to lead to robust performance. Therefore, a common trait in our methods is the hypothesis of hidden structure in the data. To this end, we employ conditional log-linear models over structures, that are firstly capable of incorporating a wide variety of features gathered from the data as well as various lexica, and secondly use *latent variables* to model missing information in annotated data. Our approaches towards solving these two problems achieve state-of-the-art accuracy on standard corpora.

For the frame-semantic parsing problem, we present fast inference techniques for jointly modeling the semantic roles of a given predicate. We experiment with linear program formulations, and use a commercial solver as well as an exact dual decomposition technique that breaks the role labeling problem into several overlapping components. Continuing with the theme of hypothesizing hidden structure in data for modeling natural language semantics, we present methods to leverage large volumes of unlabeled data to improve upon the shallow semantic parsing task. We work within the framework of graph-based **semi-supervised learning**, a powerful method that associates similar natural language types, and helps propagate supervised annotations to unlabeled data. We use this framework to improve frame-semantic parsing performance on unknown predicates that are absent in annotated data. We also present a family of novel objective functions for graph-based learning that result in sparse probability measures over graph vertices, a desirable property for natural language types. Not only are these objectives easier to numerically optimize, but also they result in smoothed distributions over predicates that are smaller in size.

The experiments presented in this dissertation empirically demonstrates that missing information in text corpora contain considerable semantic information that can be incorporated into structured models for semantics, to significant benefit over the current state of the art. The methods in this thesis were originally presented by Das and Smith (2009, 2011, 2012), and Das et al. (2010, 2012). The thesis gives a more thorough exposition, relating and comparing the methods, and also presents several extensions of the aforementioned papers.

To my family.

Acknowledgments

First and foremost, I thank my advisor Noah Smith for teaching me the range of things one can possibly aspire to learn in graduate school – perform high impact research, write research papers, teach and inspire a class of pupils, give a talk, and advise junior students. During the initial years, he closely guided me when I needed hand-holding and eventually helped me go my independent way and find my niche. I will be forever grateful to him for letting me work on a topic of my interest that constitutes a majority of this dissertation, the focus of which went beyond the boundaries of research grants.

I thank my other dissertation committee members – William Cohen, Lori Levin and Dan Roth, who provided invaluable advice. William gave me feedback and suggestions with the perspective of a machine learning researcher, that enriched the statistical techniques I adopted in my work. Lori prevented me from being oblivious to the linguistic problems that I set out to model, by being too engrossed in the statistical aspects. Several problems that I have addressed in this dissertation have been inspired by Dan's research on shallow semantic parsing of text, and he shaped many research ideas through the communication we had over the past few years. I also thank Alex Rudnicky, my Masters advisor, under whose auspices I first started performing research in statistical natural language processing; I am thankful to him for his invaluable advice to continue with the Ph.D. program in CMU in 2008.

During my stay in CMU, several current and former faculty members have kindly shared their opinions and helped me in various ways: Alan Black, Jamie Callan, Bob Frederking, Alex Hauptmann, Alon Lavie, Teruko Mitamura, Eric Nyberg, Kemal Oflazer, Bryan Routledge and Stephan Vogel. The current and erstwhile ARK research group members provided me with an outstanding research atmosphere, helped me with several research projects, taught me new things every day, and corrected me when I went wrong. Thanks to: Waleed Ammar, David Bamman, Victor Chahuneau, Desai Chen,[1] Shay Cohen,[1] Chris Dyer, Kevin Gimpel,[1] André Martins,[1] Michael Heilman,[1] Behrang Mohit, Brendan O'Connor,[1] Nathan Schneider,[1] Yanchuan Sim, Mengqiu Wang, Tae Yano and Dani Yogatama.[1] I am thankful to André, Kevin and Shay for answering many questions over the years and collaborating me on projects I am particularly proud of. Special thanks to Nathan and Tae for sharing an office with me. I am

[1]Thanks to these individuals for collaborating on projects and co-authoring research papers with me.

grateful to several other CMU students and postdocs who have helped me with my research: Satanjeev Banerjee, Sourish Chaudhuri, Jacob Eisenstein,[1] Mahesh Joshi,[1] Mohit Kumar,[1] Abhay Harpale, Ni Lao, Partha Talukdar and Le Zhao. Within the LTI staff, I thank Mary Jo Bensasi, Linda Hager, Dana Houston, Radha Rao and Stacey Young for making administrivia easy and for answering queries about visa, travel and course requirements.

I am extremely grateful to Slav Petrov,[1] my host during an internship at Google Research, for providing me with much needed research breadth; my work with him introduced me to several ideas that I have adopted in this thesis. Thanks to Ryan McDonald,[1] Shankar Kumar and Amar Subramanya for their help and encouragement during the internship and afterwards; post completion of my Ph.D., I will be joining them and the New York research group to continue the productive collaboration that we started two years back. I am also thankful to several faculty members and researchers from other universities and research labs for helpful discussions: Collin Baker, Chris Brockett, Jason Eisner, João Graça, Rebecca Hwa, Sanjeev Khudanpur, Kevin Knight, Daniel Marcu, Mausam, David McAllester, Andrew McCallum, Sujith Ravi, David Smith, Mark Steedman, Lucy Vanderwende, Bonnie Webber, David Yarowsky and Luke Zettlemoyer. Special thanks go to Monojit Choudhury[1] and Pradipta Ray, who introduced me to NLP when I was a second year undergraduate student.

My six year stay at Carnegie Mellon and Pittsburgh has been a memorable experience because of several individuals. Thanks to Abhaya Agarwal, Sachin Agarwal, Shilpa Arora, Sameer Badaskar, Suresh Ganapathy, Abhay Harpale, Mahesh Joshi, Anagha Kulkarni and Usha Kuppuswamy for making life enjoyable during the first few years in Pittsburgh. I extend my heartfelt gratitude to Pradipta Ray for setting up appointments with faculty when I wanted to transfer to CMU back in 2006, discussing research problems and spending countless nights conversing over music, cinema, politics and literature. Thank you to Sourav and Sudarshana Bhattacharya, and Tathagata and Soumitri Dasgupta, for being like family several thousand miles away from home.

I am grateful to my brother Nilanjan Das, who recently started his Ph.D. at MIT, for providing me with true academic inspiration since he was very young. I am deeply thankful to my parents Aditi and Dipak Das for giving me unconditional support during my dissertation work. My father finished a Ph.D. after growing up in abject poverty as a refugee in suburban Bengal. His academic resolve, irrespective of the impediments he faced, has implicitly inspired me when I doubted myself. I am grateful to my uncle and aunt Dilip and Sathi Das for their unending love and encouragement. I thank Indrani and Tapan Chaki, for their emotional support, for believing in me ever since they first met me several years back when I was a junior graduate student. I am thankful to them for welcoming me into their family and for being proud of me that I got here eventually.

Finally, I thank my wife Rohini Chaki, who through her selfless friendship and love shaped my life during this work and filled the last six years with laughter and joy. Her

resolve helped us overcome the early years of unbearable intercontinental separation; she accepted my unreasonable demands and made unconditional sacrifice to be with me in Pittsburgh. From paper rejections and frequent failed ideas to an unexpected working experiment and getting the first citation, she experienced the frequent angst and the occasional glory of graduate school with me. One cannot attribute the achievement associated with this dissertation to me alone – I share it with her.

The material in this dissertation is based upon work supported by the National Science Foundation under Grant No. 0844507, the Defense Advanced Research Projects Agency under Grant No. NBCH-1080004, Google (the "Worldly Knowledge Project") and the Qatar National Research Foundation under Grant No. NPRP08-485-1-083. The author is also grateful to Yahoo! and the Pittsburgh Supercomputing Center (NSF Grant No. TG-DBS110003) for providing computational resources. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the aforementioned organizations.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Wide-coverage semantic analysis of text is currently an obstacle for robust natural language understanding. Broadly, semantic analysis of text thus far has considered the conversion of text into deep structures such as **logical forms** using training corpora belonging to a narrow domain (Ge and Mooney, 2005; Zettlemoyer and Collins, 2005; Liang et al., 2011), or **semantic role labeling** that investigates predicate-argument structures of verbs and nominal items producing shallow symbolic output (Palmer et al., 2005). While the former suffers from the lack of coverage because the supervised training methods are limited to very small corpora, the latter assumes a small inventory of argument labels to gather sufficient amount of training data, resulting in inconsistency among the meaning of the labels across different semantic frames (Yi et al., 2007). **Word sense disambiguation** is another popular task (Brown et al., 1991; Yarowsky, 1995) whose goal is to identify the correct meaning of a word given its context. However, disambiguating word meaning does not result in predicate argument structures, which are useful semantic representations.

Among various other attempts to model natural language semantics, one major goal has been to discover **semantic relationships between sentence-pairs**, mostly investigated via the problem of recognizing textual entailment (Dagan et al., 2005; Bar-Haim et al., 2006; Giampiccolo et al., 2007). Most research in this area has either resorted to the use of shallow bag-of-words based classifiers that leads to robustness but fails to model structural correspondences between sentence pairs that govern a semantic relationship (Corley and Mihalcea, 2005; Glickman et al., 2005), or have modeled these sentential relationships using brittle forms of logical inference that do not generalize to varied domains of text (Bos and Markert, 2005; MacCartney and Manning, 2007) primarily because these models are trained on corpora from restricted domains.

In this dissertation, we investigate structured models for natural language semantics: specifically we focus on recognizing the **paraphrase** relationship between two sentences and semantic analysis of text in the form of **frame-semantic parsing**. We hypothesize that the semantic analysis of a natural language utterance is closely related to its syntax, and exploit useful syntactic representations to this end. We also leverage lexical resources in our models, to incorporate expert knowledge in our methods. In most of our models, we apply a probabilistic framework as it suits our needs with respect to model combination and the ease of building feature-rich models. A common trait in our work is the joint modeling of substructures, often using fast inference techniques that obviates naïve independence assumptions present in prior work.

A common bottleneck across all semantic analysis tasks is the absence of richly annotated corpora. To cite a few examples, lexical resources used to assist semantic analysis are often scarce, word aligned corpora for sentence-pair relationships are few, and large corpora of sentences annotated with semantic structures are limited. To sidestep the dearth of annotated data, we model **latent structure** in data for both the tasks in consideration. Finally, we use large volumes of unlabeled data to generalize our models to unknown lexical items and perform **semi-supervised learning** to demonstrate that useful semantic information for analysis can be extracted from raw text. Before delving into the details of our methods in the following chapters, we will provide a brief background on statistical modeling of natural language semantics and motivate the necessity of semantic analysis of text.

1.1 Statistical Methods in NLP

The past two decades have witnessed an empirical revolution in natural language processing (NLP). The area has increasingly been influenced by machine learning techniques and statistical modeling of natural language phenomena has evolved to be the well-accepted norm. The availability of the Penn Treebank (Marcus et al., 1993) led to statistical models of natural language *syntax* in the form of probabilistic context free grammar variants, the more famous manifestations being the Charniak and the Collins parsers (Charniak, 2000; Collins, 2003). Since then, treebanks for several other languages have been built, resulting in robust syntactic parsers, both for phrase-structure and dependency grammars. Data-driven methods for other NLP tasks like text chunking (Tjong Kim Sang and Buchholz, 2000), named-entity recognition (Tjong Kim Sang, 2002), coreference resolution (Grishman and Sundheim, 1995) and machine translation (Brown et al., 1993), have motivated the ubiquitous use of empirical methods in natural language analysis.

Among various empirical methods, *probabilistic* modeling of language structure has been a popular form, because the probabilistic genre allows a flexible framework with several advantages. These models facilitate the combination of simpler models, promote the use of overlapping features (in log-linear models), and can accommodate latent variables to model unseen structure. Semi-supervised extensions of supervised probabilistic models are intuitive and have commonly been used in NLP. In recent times, probabilistic models have been widely used in syntactic parsing (Petrov et al., 2006; Smith and Smith, 2007; Petrov and Klein, 2008), sequence labeling tasks (Finkel et al., 2005), grammar induction (Smith, 2006) and machine translation (Koehn et al., 2007).

Probabilistic modeling for natural language *semantics* has also been popular. For example, significant amount of work on modeling lexical semantics exists, and has been popular: a vast proportion of research on word sense disambiguation (Brown et al., 1991; Bruce and Wiebe, 1994; Yarowsky, 1995) and creation of lexical resources (Snow et al., 2006; Haghighi et al., 2008) have made use of such models. Recent research on shallow semantic parsing in the form of semantic role labeling (SRL) has largely exploited probabilistic modeling (Gildea and Jurafsky, 2002; Cohn and Blunsom, 2005). Several lines of work on natural language inference or textual entailment have made use of probabilistic models to determine whether semantic relationships between sentence pairs exist (Glickman et al., 2005; Glickman and Dagan, 2005).

In this work, we will widely employ probabilistic methods for tasks in computational semantics. We will observe why probabilistic methods suit the tasks at hand, how these methods assist in modeling structures unobserved in supervised data, and how unlabeled text can be brought into probabilistic modeling with the hope of extracting useful semantic information from text.

1.2 Why Computational Semantics?

Semantics deals with the literal representation of meaning in natural language utterances. Computational modeling of semantics is essential for deeper understanding of natural language, and in this section, we motivate the necessity of automatic semantic analysis of text.

1.2.1 Deep Natural Language Understanding

To fully automate natural language understanding, representations beyond popular formalisms such as dependency and phrase-structure grammars that model syntax are necessary. Consider the sentence in Example 1.1:

(1.1) Marco Polo wrote an account of Asian society during the 13th century.

Figure 1.1(a) shows an example phrase-structure parse that uses Penn Treebank (Marcus et al., 1993) conventions, while Figure 1.2 shows an example dependency syntax tree for the same sentence. The dependency tree is *labeled* in that the head-modifier relations are marked by a handful of syntactic relations. State-of-the art parsers like the Collins

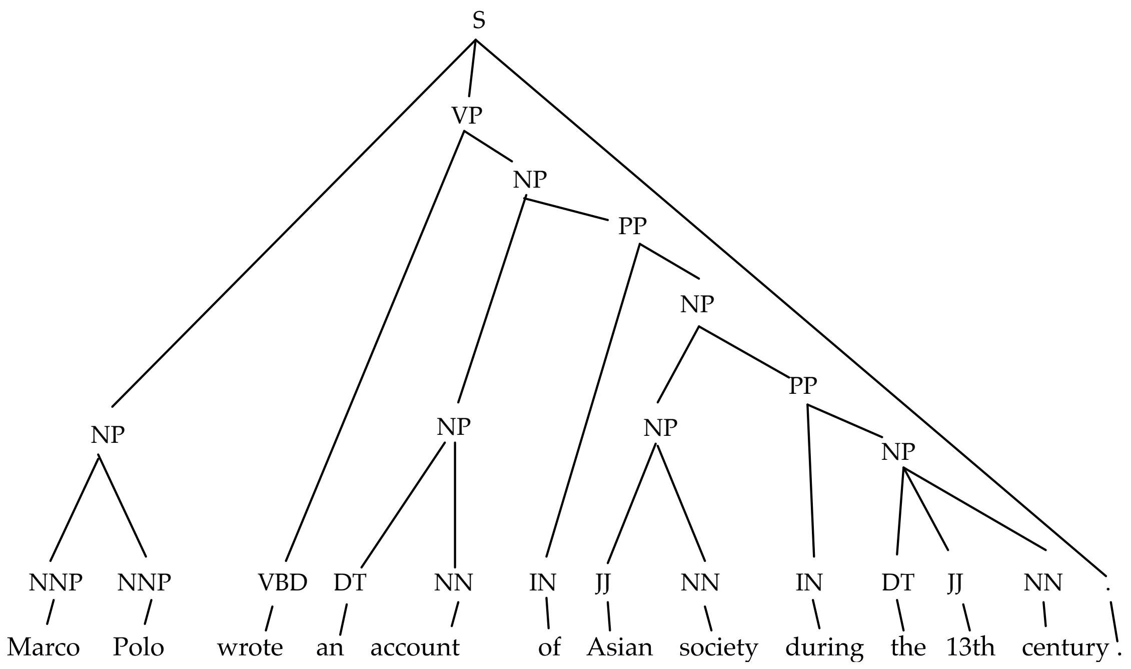

S
VP
NP
PP
NP
PP
NP
NP
NP
NP
NNP NNP VBD DT NN IN JJ NN IN DT JJ NN .
Marco Polo wrote an account of Asian society during the 13th century .

Figure 1.1: A Penn Treebank style phrase-structure syntax tree for Example 1.1.

(2003) and the Charniak (2000) parsers produce parses similar to Figure 1.1 while parsers such as the MST parser (McDonald et al., 2005), the Malt parser (Nivre et al., 2004), or stacked dependency parsers (Martins et al., 2008) or would produce an analysis such as Figure 1.2.

Although such syntactic representations have proved to be very useful for several applications such as machine translation (Zollmann and Venugopal, 2006), question answering (Wang et al., 2007) and relation extraction (Culotta and Sorensen, 2004), phenomena such as sense ambiguity or semantic frames of lexical items in a sentence are not analyzed by plain syntax. For example, consider the word "account" in Example 1.1. From the parses shown in Figures 1.1 and 1.2, it is unclear to a computer system whether the word means "a description of facts, conditions, or events" or "a statement of transactions during a fiscal period and the resulting balance."[1]

Figure 1.3 on the other hand portrays a *semantic* analysis of the same sentence, following the paradigm of frame semantics (Fillmore, 1982). We will go into the details

[1]See `http://www.merriam-webster.com/dictionary/account` for more dictionary definitions of the word "account."

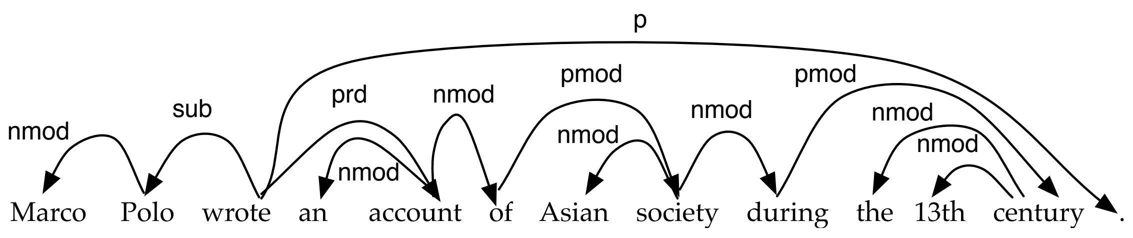

p
nmod
sub
prd
nmod
nmod
pmod
nmod
nmod
pmod
nmod
nmod
Marco Polo wrote an account of Asian society during the 13th century .

Figure 1.2: A labeled dependency syntax tree for Example 1.1.

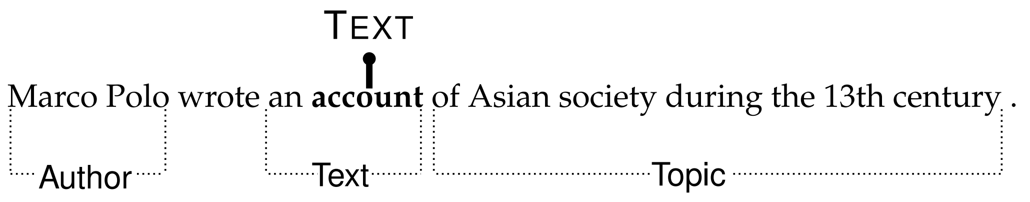

TEXT
Marco Polo wrote an account of Asian society during the 13th century .
Author
Text
Topic

Figure 1.3: A frame-semantic parse for Example 1.1.

of frame semantics in Chapter 5, but essentially the frame-semantic parse of a sentence results in a collection of semantic frames evoked by words or phrases in a sentence, and for each frame, a set of semantic roles are also predicted. In the parse shown in Figure 1.3, only one semantic frame is evoked, by the capitalized word "account". The semantic frame in this case is TEXT. It has three semantic roles Author, Text, and Topic as marked under token spans in the figure. Unlike a syntactic parse of a sentence, this analysis clearly portrays the fact that the word "account" is a form of text that has an author and a particular topic, and not a record of transactions. In fact, these word senses can be derived from a lexical resource like FrameNet (Fillmore et al., 2003), that lists words and phrases with various semantic frames they can evoke, along with each frame's possible set of semantic roles.

1.2.2 Semantics in NLP Problems

What applications can benefit from this type of semantic analysis? One can cite several; however, let us choose the popular task of machine translation. To test the efficacy of a state-of-the-art machine translation system for a low-resource language pair, we provided the sentence in Example 1.1 to the English to Hindi translation engine of Google.[2] Unfortunately, the Hindi translation of the sentence produced by this system is the fol-

[2]See `http://translate.google.com/`.

lowing:

(1.2)

मार्को	पोलो	१३	बीं	सदी	के	दौरान		
Marco	Polo	13	th	century	of	during		
एशियाइ	समाज	के	एक	खाते	में	लिखा	था	।
Asian	society	of	one	record of transactions	in	wrote	had	.

In this Hindi sentence with English glosses, the literal translation of the underlined word खाते is "a record of transactions," which indeed is another meaning of the word "account," however not in the context of Example 1.1. Moreoever, an extraneous postposition में after खाते is introduced, which changes the core meaning of the translation, resulting in the following literal translation:

(1.3) Marco Polo wrote in the Asian society's record of transactions during the 13th century.

It is easy to point out that the correct Hindi word for "account" was not used by the translation system, which possibly did not encounter the desired word sense of "account" in its training data. To exemplify the necessity of semantic analysis of text, we next presented the following sentence to the same translation system:

(1.4) Marco Polo wrote a chronicle of Asian society during the 13th century.

This sentence is a straightforward modification of Example 1.1, with "an account" replaced by "a chronicle". The replacement roughly retains the meaning of the sentence, and like "account", the word "chronicle" belongs to the same semantic frame TEXT according to the FrameNet lexicon. Example 1.5 is the Hindi translation produced by the system, which translates "chronicle" to the underlined word इतिहास, meaning "history", resulting in the desired meaning. The possessive marker introduced in Example 1.2 is also absent, making it an acceptable translation.

(1.5)

मार्को	पोलो	१३	बीं	सदी	के	दौरान	
Marco	Polo	13	th	century	of	during	
एशियाइ	समाज	का	इतिहास	लिखा	था	।	
Asian	society	of	history	wrote	had	.	

Semantic analysis of the English side could possibly have avoided the scenario presented above. A frame-semantic parse as in Figure 1.3 of the sentence in Example 1.1 would tag the word "account" with the semantic frame TEXT, which would have provided a signal to the translation system indicating that the desired sense of the word in the target side should conform to the same semantic frame.

Previous researchers have incorporated semantic analysis of text into various applications and have reported success. Bilotti et al. (2007) used semantic roles to improve question answering. Their conclusions suggest that semantic processing of web documents can produce results more relevant to input questions. They used PropBank (Kingsbury and Palmer, 2002) semantic role labeling to preprocess web data used for retrieval, followed by clever indexing. A blend of syntax and lexical semantics was used for question answering by Wang et al. (2007), where lexical similarity in the form of WordNet (Fellbaum, 1998) lookups were leveraged to rank candidate answers to questions. Shen and Lapata (2007) used FrameNet semantic structures to improve question answering; their approach treated the answer selection problem as graph matching, where the graphs incorporated semantic information. Qiu et al. (2006) used semantic roles to improve paraphrase identification. Predicate-argument tuples were matched between candidate sentence pairs to detect a paraphrase relationship. Das et al. (2008) have leveraged semantic roles for template creation for abstractive summarization in closed domains. Their technique involved clustering of human written summaries using a similarity metric based on semantic roles. In recent work, semantic roles have been used in statistical machine translation by Wu and Fung (2009). They used a two pass model where the first pass was a typical phrase-based approach, while the second pass was used to develop a re-ordering strategy using semantic role annotations. Their preliminary experiments resulted in improvements in translation quality measured by the BLEU score (Papineni et al., 2001). Recently, features extracted from a PropBank semantic role labeler have been used in a tree-to-string transducer model to improve fluency of automatic translation (Liu and Gildea, 2010).

A separate line of work in computational semantics has looked at relationships between pairs of sentences. A vast body of research has been performed in recognizing textual entailment (RTE), where the goal is to identify whether a hypothesis is entailed by a premise.

(1.6) In 1998, the General Assembly of the Nippon Sei Ko Kai (Anglican Church in Japan) voted to accept female priests.

(1.7) The Anglican Church in Japan approved the ordination of women.

Examples 1.6 and 1.7 constitute a sentence pair where the second sentence is entailed by the first. Determining whether the meaning of one sentence is implied by another has been compared to the Turing test (Bos and Markert, 2005), as it may require deep semantic understanding of language. This is exemplified by the pair of sentences presented above. For example, the fact that "ordination" and "accepting female priests" are embodiments of the same meaning requires deep semantic analysis of text. Other examples of such relationships include equivalence or paraphrase (Dolan and Brockett, 2005, two sentences conveying the same information), and contradiction (de Marneffe et al., 2008, the pair providing contrasting information).

Modeling semantic relationships between pairs of sentences is relevant for various NLP applications like multi- document summarization, large news clustering systems that need to better understand standpoints of different news sources,[3] improved question answering (Harabagiu and Hickl, 2006) or automatic grading of student responses given reference answers. These systems use methods for finding an entailment relationship to model answers to a given question and a collection of text. Modeling of phrasal paraphrases have led to improvements in statistical machine translation for low-resource scenarios (Callison-Burch et al., 2006; Marton et al., 2009). Very recently, Padó et al. (2009) have used features motivated by textual entailment to produce better machine translation evaluation metrics, in comparison to traditional and popular bag-of-words metrics like BLEU (Papineni et al., 2001).

Semantic processing of text is essential from two standpoints. First, wide-coverage and robust natural language understanding can be furthered only through better semantic processing of text, in the forms of lexical semantics, parsing or through the modeling of relationships between sentences. Second, a variety of NLP applications still need improvement, and better semantic understanding of text will directly help towards that goal.

1.3 Contributions of the Thesis

As described at the onset of this chapter, we investigate statistical models for two semantic analysis tasks: **paraphrase identification** and **frame-semantic parsing**. Although these problems have been addressed by the NLP community before, the described research departs from previous work in several dimensions. Our major contributions are:

1. We model complex semantic phenomena using structured statistical models, instead of relying on a collection of naïve classifiers. Joint modeling of substructures relevant to a problem is done by efficient, and at times approximate inference techniques. To this end, whenever possible, we make use of distributed computing to facilitate fast parameter estimation and inference.

2. Large quantities of data containing rich semantic annotations do not exist. We attempt to model unseen structure in the data by employing latent variables in our probabilistic models. While unseen during the estimation and the testing phases, meaningful latent structure can be uncovered as a by-product of MAP inference. In these scenarios, we make use of a probabilistic framework for elegant modeling of latent variables.

[3]See `http://www.ark.cs.cmu.edu/RAVINE`

3. Our models for the two semantic analysis problems result in state-of-the-art performance on standard corpora. Additionally, we have publicly released our frame-semantic parser for the NLP community to use as an open-source project.[4]

4. We use large amounts of unlabeled data for improving the coverage of our frame-semantic parser. The supervised version of our parser is not able to accurately handle predicates previously unknown in supervised data. We use a graph-based semi-supervised learning framework to improve the parser's coverage on unseen predicates.

5. Finally, we present a family of graph-based learning algorithms that are easy to optimize and produces sparse distributions over labels on the graph vertices. This desirable property, which is suitable to language processing problems, not only results in better results than the state of the art, but also results in sparser distributions that require less storage space.

1.4 Organization of the Thesis

The thesis is organized as follows.

- Chapter 2 focuses on relevant scientific work on semantic relationships between sentence pairs, with a focus on modeling paraphrase. Next, it describes relevant work on shallow semantic parsing, especially frame-semantic analysis of text. Herein, we contrast our techniques with previous work and other forms of semantic parsing tasks and methods.

- Chapter 3 describes a set of tools used in this thesis. Examples of these tools are syntactic representations, probabilistic log-linear models and parallelization schemes useful for numerical optimization techniques that are widely used in this thesis.

- Chapter 4 investigates our model for recognizing a paraphrase relationship between two sentences. We describe our probabilistic technique that we design for the problem, and the experiments and the results achieved on a popular corpus. We also compare and analyze our results in comparison to related approaches. The material presented in this chapter is an exposition of Das and Smith (2009).

- Chapter 5 describes in detail the model used for frame-semantic parsing. It describes the task in detail, the lexicon used to derive expert knowledge, the structured model that solves the task, and various fast inference techniques used for

[4]See `http://www.ark.cs.cmu.edu/SEMAFOR`.

parsing raw text. We present results on a standard benchmark dataset for comparison with previous work, and also present results on a more recent larger dataset. This chapter is an extension of Das et al. (2010, 2012).

- Chapter 6 consists of a semi-supervised extension of the frame-semantic parser; in this chapter, we describe a method of extending the coverage of the frame-semantic parser on predicates unseen in annotated data. To this end, we use a graph-based semi-supervised learning approach to model unseen predicates. We also present alternative graph-based learning algorithms that result in sparser distributions over graph vertices, are easy to optimize and result in models that are significantly smaller in size. This chapter is based on Das and Smith (2011, 2012).

- Finally, Chapter 7 concludes the findings of this dissertation and provides future directions of research.

Chapter 2

Literature Review

This chapter reviews previous scientific work on computational semantics relevant to the two broad problems that we investigate. §2.1 looks at techniques used in modeling semantic relationships between a sentence pair, and focuses on the recognition of sentential paraphrases. §2.2 reviews relevant research on shallow semantic parsing, especially focusing on analysis based on frame semantics (Fillmore, 1982).

2.1 Models of Sentence-sentence Relationships

In recent years, modeling semantic relationships between sentence pairs has generated considerable interest in the NLP community. In this section, we will review relevant previous work in textual entailment, and paraphrase identification.

2.1.1 Recognizing Textual Entailment

Among various relationships like entailment, paraphrase and contradiction, the first has been of specific interest to a large fraction of the community. Dagan et al. (2005), Bar-Haim et al. (2006) and Giampiccolo et al. (2007) organized the first three Recognizing Textual Entailment (RTE) shared tasks where several participants built models for textual inference. In recent years, the Text Analysis Conference (TAC)[1] has continued to organize the RTE challenges. As mentioned in Chapter 1, the RTE task essentially asks whether there exists an entailment relationship between a premise and a hypothesis. A popular version of the task is a binary classification problem, where a system needs to predict whether there exists an entailment relationship or not. Another version presents a three-way classification task where the relationships can be either entailment, non-entailment or "unknown".

[1]See `http://www.nist.gov/tac/`.

Example 2.1 and 2.2 is a premise-hypothesis pair taken from the RTE3 challenge, and is one where the entailment relationship holds.

(2.1) ”The Extra Girl” (1923) is a story of a small-town girl, Sue Graham (played by Mabel Normand) who comes to Hollywood to be in the pictures. This Mabel Normand vehicle, produced by Mack Sennett, followed earlier films about the film industry and also paved the way for later films about Hollywood, such as King Vidor's ”Show People” (1928).

(2.2) ”The Extra Girl” was produced by Sennett.

It is noticeable that the premise often is quite long, containing multiple sentences, and the hypothesis is short. The contents of the hypothesis sentence in this example can be inferred from the premise by first preprocessing it with tools like named-entity recognition and coreference resolution systems, aligning relevant phrases across the two utterances, and finally using an inference step. Identification of textual inference thus becomes non-trivial. The following is another pair taken from the same dataset:

(2.3) Take consumer products giant Procter and Gamble. Even with a $1.8 billion Research and Development budget, it still manages 500 active partnerships each year, many of them with small companies.

(2.4) 500 small companies are partners of Procter and Gamble.

Clearly this pair is one where the premise does not imply the contents of the hypothesis, and getting to this decision for a state-of-the-art NLP system is hard because it needs semantic analysis and logical inference stages.

Broadly, two kinds of approaches have been used to model RTE. First, simple bag-of-words classifiers have been employed to predict the classes. Glickman and Dagan (2005), Jijkoun and de Rijke (2005) and MacCartney et al. (2006) describe bag-of-words models employing lexical and semantic overlap between the two sentences to predict the entailment relationship. These approaches do not model any form of structural correspondence between the premise and the hypothesis, but are robust and generally work well for a considerable proportion of sentence pairs. However, complex effects of antonymy, variation of predicate-argument structure and negation are not captured by these models. Another line of work has looked at deep analysis of the sentences to result in logical forms. Bos and Markert (2005) used deep semantic analysis to produce logical forms for the premise and the hypothesis and applied a theorem prover to find textual entailment. This method resulted in high precision, but suffered from poor coverage on the RTE1 test set. In a more recent approach, MacCartney and Manning (2007) used a less strict formalism called Natural Logic, where lexical items in the premise and the hypothesis were first aligned, and then local entailment decisions were taken using a classifier that incorporated several lexical, syntactic and semantic features. The local decisions were joined using compositional rules, to result in a global entailment decision. This system had very high precision and a combination with a simpler overlap based model resulted in good performance on the RTE datasets.

Recently, in another line of work, there have been efforts to model wide coverage subsentential entailment rules that could be potentially used in an entailment system; Berant et al. (2011) provide an example of this line of research. In this work, the authors attempt to learn many entailment rules globally, using a graph structure. The graph contains predicates, and the edges correspond to entailment relationships. Using the graph structure, new entailment rules are induced, that have high recall over unrestricted text

2.1.2 Paraphrase Identification

In our work, we are interested in a different but related sentence-pair relationship, that of **paraphrase**. The paraphrase relationship between two sentences can be thought of *bidirectional entailment*. Modeling the paraphrase relationship between sentences is not new. To our knowledge, the first work in this area was presented by McKeown (1979), who described a system that paraphrased user queries to a natural language computer interface to ensure that the system understood the user correctly. Since then, there has been a large body of work on automatic generation or extraction of paraphrases. Ravichandran and Hovy (2002), Barzilay and Lee (2003) and Dolan and Brockett (2005) have presented data-driven techniques for finding sentential paraphrases. In summary, these approaches looked at large amounts of raw text and used surface level similarity to extract similar meaning sentences.

At finer granularity levels, finding synonyms of *words* and paraphrases of *phrases* has been investigated by another section of the community. Distributional similarity-based methods have been investigated by Pereira et al. (1993) and Lin and Pantel (2001) where monolingual corpora were processed to gather syntactic contexts of words, and common contextual information was used to cluster similar meaning words. A series of work has followed, with a recent one attempting to cluster phrases from a web-scale text corpus (Lin and Wu, 2009). Other approaches to find semantically equivalent phrases have attempted to harvest multiple translations of the same foreign source (Barzilay and McKeown, 2001). Using large volumes of multilingual corpora to extract phrasal paraphrases has been the most recent and attractive avenue of research. Bannard and Callison-Burch (2005) presented a probabilistic method of finding paraphrases from bilingual parallel corpora. From a given sentence pair in the parallel corpora, they chose a phrase in the source language, and found its translation phrase in the target language. This phrase in the target language was fixed as a pivot. Next they scanned for this pivot phrase in the other sentence pairs in the corpora, and found several translations in the source language. These translations were deemed to be potential paraphrases of the original source phrase. This approach worked quite well, and recent extensions Callison-Burch (2008); Kok and Brockett (2010) have further improved paraphrasing quality.

In our work, rather than the generation or extraction of paraphrases from free text, we are concerned with the problem of recognizing the paraphrase relationship in a sentence pair. Examples 2.5 and 2.6 belong to a pair that essentially talk about the failing revenue of a company, and is a paraphrase pair:

(2.5) Revenue in the first quarter of the year dropped 15 percent from the same period a year earlier.

(2.6) With the scandal hanging over Stewart's company, revenue in the first quarter of the year dropped 15 percent from the same period a year earlier.

Above, the second sentence has an extra clause in the beginning which is absent in the first sentence; however, since the salient semantic content lies in the remainder of the sentence and this content is essentially the same as the first sentence, the pair was judged as paraphrase. From this example, it is noticeable that the annotators took liberty in their decision making in that they did not label pairs that *strictly* contain the same information as the sole positive examples. In context of our benchmark corpus, this loose definition makes the paraphrase identification task hard.

Another pair that has similar lexical content, but are not equivalent in meaning is cited below. The first sentence in the pair contains some information about the police searching for traps, which is absent in the second sentence; the annotators considered the extra content of the first sentence to be important, making the pair a non-paraphrase example.

(2.7) Security lights have also been installed and police have swept the grounds for booby traps.

(2.8) Security lights have also been installed on a barn near the front gate.

The paraphrase identification task is a binary classification problem where a given pair of sentences need to be labeled as paraphrase or not. Data-driven techniques for this task has mostly leveraged the Microsoft Research Paraphrase Corpus (Dolan et al., 2004; Quirk et al., 2004, MSRPC) to build models of paraphrase. Like the textual entailment task, this task also has witnessed two major genres of modeling approaches: using bag-of-words feature based classification, and methods involving deep lexical, syntactic and semantic processing of the individual sentences in the pair.

Among the first category of work on paraphrase identification, Zhang and Patrick (2005) used text canonicalization to transform each sentence in the pair into a simplified form, e.g. by changing passive voice to active voice and by changing complex future tense phrases to simpler ones. Next, they used a classifier trained on lexical match between the canonicalized sentences to predict the paraphrase relationship. Corley and Mihalcea (2005) used word-to-word lexical similarity to measure the similarity of two sentences in a given pair. Another line of work used several surface level features like lexical overlap, overlap of syntactic dependencies in the two sentences, the BLEU score between the two sentences and the difference in sentence lengths to train a discriminative classifier for the paraphrase relationship (Finch et al., 2005; Wan et al., 2006; Malakasiotis, 2009). Wan et al. (2006) specifically used a Support Vector Machine (Vapnik, 1995, SVM henceforth) to train their model, and we use their system as a strong baseline for comparison. Qiu et al. (2006) used semantic role labeling to find dissimilarity between sentences, and used an SVM to classify whether two sentences are paraphrases of each other.

In contrast to all the aforementioned work on recognizing paraphrases, we model the problem as a monolingual translation scenario, where we assume that one sentence in the pair has been transformed into the other using a loose syntactic generative process, defined by a *quasi-synchronous grammar* (Smith and Eisner, 2006). This process, which is probabilistic, gives us a posterior probability that indicates whether the pair is a paraphrase or not. We combine dependency syntax and lexical semantics as WordNet lookups in a single model. Word alignments are also modeled in our method, and are treated as latent variables and are marginalized out.

Recently, Chang et al. (2010) have presented a generic discriminative technique for modeling relationships between sequences with constrained latent representations. As a specific application of their model, they modeled the paraphrase identification problem and also used latent alignments as intermediate structures that they jointly model with the binary decision problem. Their model, trained using a margin-based online learning algorithm, namely "Learning over Constrained Latent Representations" performs well across several applications; however the setup is not probabilistic, and does not present an analogy with monolingual translation, which forms the basis of our model. Heilman and Smith (2010) use a variant of tree edits to transform a syntax tree of one sentence to another, and incorporate the edit operations as features in a logistic regression model. This work comes very close to our method, but does not model word alignments explicitly. Most related to our approach, Wu (2005) used inversion transduction grammars—a synchronous context-free formalism (Wu, 1997)—for this task. Wu's model can be understood as a strict hierarchical maximum-alignment method. In contrast, our alignments are soft (we sum over them), and we do not require strictly isomorphic syntactic structures. Most importantly, our approach is founded on a stochastic generative process and estimated discriminatively for this task from data, while Wu presented a rule-based system, whose robustness on a variety of sentence pairs is questionable; we outperform Wu's approach on the MSR paraphrase corpus by a significant margin.

Finally, Socher et al. (2011) present a powerful neural network model for learning the paraphrase relationship. They use a large unlabeled corpus to learn embeddings of words and also represent the nodes in parse trees of a sentence in a vector space; this information is used in detecting the binary paraphrase relationship between sentence pairs. Various aspects of this work is similar to our work where relationships between parse subtrees are modeled. On the same dataset on which we measure performance, this approach outperforms our grammar based method by a small margin.

2.2 Techniques in Shallow Semantic Parsing

In this subsection, we will focus on previous scientific work relevant to the problem of frame-semantic parsing. First, we will briefly discuss work done on PropBank-style semantic role labeling, following which we will concentrate on the more relevant problem of labeling frame-semantic roles. Next, we review previous work that has used semi-supervised learning for shallow semantic parsing. Finally, we compare the frame-semantic parse representation with deeper semantic parsing formalisms, which has been a popular research focus within the computational linguistics community.

2.2.1 Semantic Role Labeling

Since Gildea and Jurafsky (2002) pioneered statistical semantic role labeling, there has been a great deal of computational work using predicate-argument structures for semantics. The development of PropBank (Kingsbury and Palmer, 2002), followed by CoNLL shared tasks on semantic role labeling (Carreras and Màrquez, 2004; Carreras and Màrquez, 2005) boosted research in this area.

Figure 2.1 shows a sentence annotated with semantic roles, taken from PropBank. PropBank annotations are closely tied with syntax, because the dataset is essentially the phrase-structure syntax trees from the *Wall Street Journal* section of the Penn Treebank (Marcus et al., 1993) annotated with predicate-argument structures for verbs. In Figure 2.1, the syntax tree for the sentence is marked with various semantic roles. The two verbs in the sentence "created" and "pushed" are the predicates. For the former, the constituent "more than 1.2 million jobs" serves as the semantic role ARG1 and the constituent "In that time" serves as the role ARG-TMP. Similarly for the latter verb, roles ARG1, ARG2, ARGM-DIR and ARGM-TMP are shown in the figure. PropBank defines roles ARG0 to ARG5 which behave in a specific manner for a given verb, and additionally defines auxiliary roles ARGM-*, examples of which are ARGM-TMP and ARGM-DIR as shown in Figure 2.1. Therefore the total number of tags in PropBank is few, and the training dataset has $\sim$40,000 sentences, thus making the semantic role labeling task an attractive one from the perspective of machine learning.

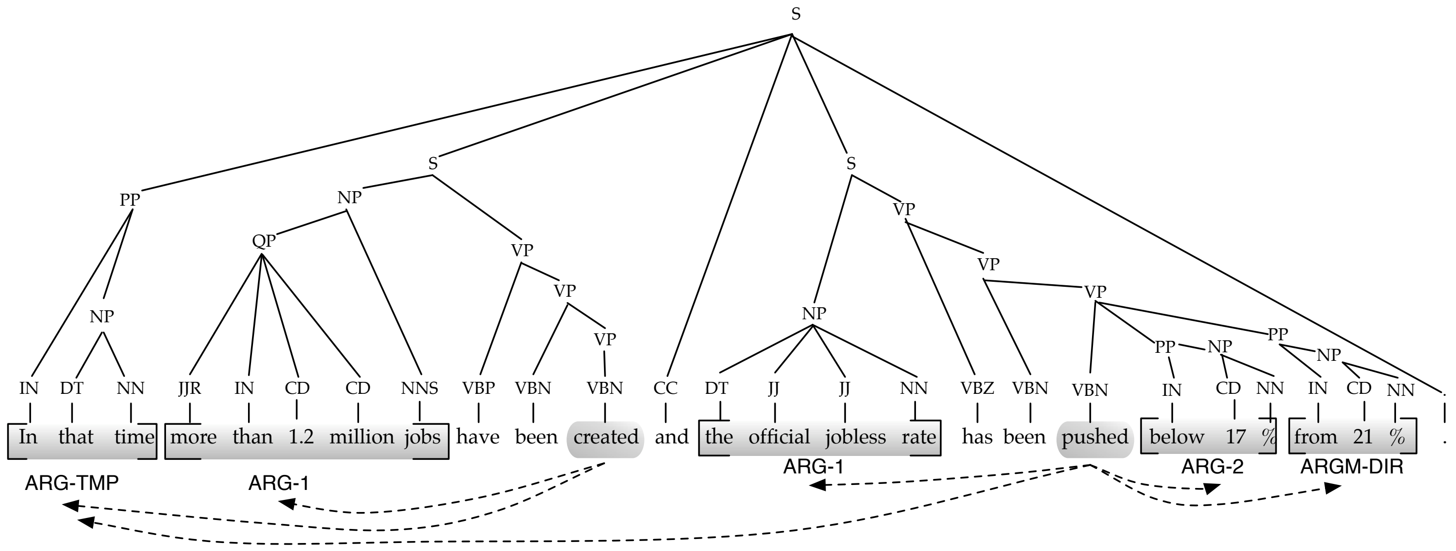

S
PP
NP
S
NP
QP
VP
VP
VP
S
VP
NP
VP
VP
PP
NP
PP
NP
IN DT NN JJR IN CD CD NNS VBP VBN VBN CC DT JJ JJ NN VBZ VBN VBN IN CD NN IN CD NN .
In that time more than 1.2 million jobs have been created and the official jobless rate has been pushed below 17 % from 21 % .
ARG-TMP
ARG-1
ARG-1
ARG-2
ARGM-DIR

Figure 2.1: A phrase-structure tree taken from the Penn Treebank (Marcus et al., 1993), and annotated with predicates and corresponding semantic roles in the PropBank (Kingsbury and Palmer, 2002). There are two verbs marked in rounded rectangles- "created" and "pushed" that serve as the predicates in this sentence. The corresponding semantic roles marked within shaded brackets are connected to the verbs using dotted arrows.

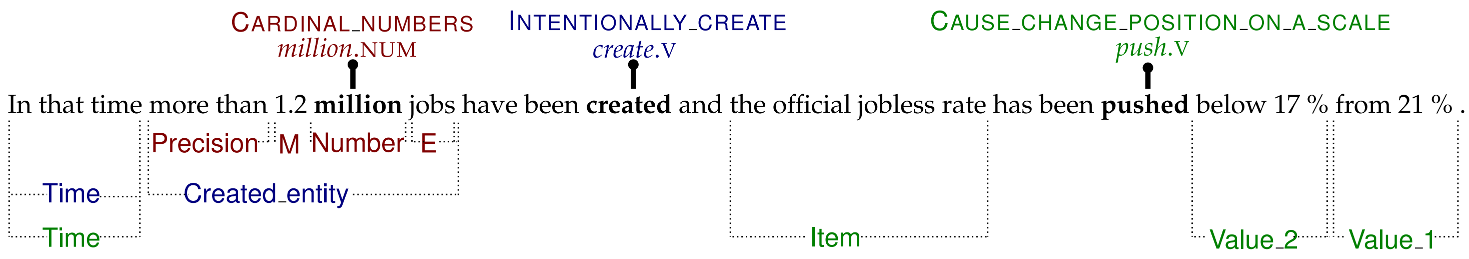

CARDINAL_NUMBERS
million.NUM
INTENTIONALLY_CREATE
create.V
CAUSE_CHANGE_POSITION_ON_A_SCALE
push.V
In that time more than 1.2 million jobs have been created and the official jobless rate has been pushed below 17 % from 21 % .
Precision
M
Number
E
Time
Created_entity
Time
Item
Value_2
Value_1

Figure 2.2: A partial depiction of frame-semantic structures for the same sentence as in Figure 2.1. The words in bold correspond to targets, which evoke semantic frames that are denoted in capital letters. Above each target is shown the corresponding lexical unit, which is a lemma appended by a coarse part-of-speech tag. Every frame is shown in a distinct color; each frame's arguments are annotated with the same color, and are marked below the sentence, at different levels. For the CARDINAL_NUMBERS frame, "M" denotes the role Multiplier and "E" denotes the role Entity.

There are many instances of influential work on semantic role labeling using PropBank conventions. Pradhan et al. (2004) present a system that uses SVMs to identify the arguments in a syntax tree, which can serve as semantic roles, followed by the classification of the identified arguments into role labels. The used several binary SVMs and used them to choose the best role. Punyakanok et al. (2004) describe a semantic role labeler that uses integer linear programming for inference and uses several global constraints to find the best suited predicate-argument structures. Joint modeling for semantic role labeling using discriminative log-linear models is presented by Toutanova et al. (2005), where the authors used global features looking at all arguments of a particular verb together in a dynamic programming and reranking framework. The *Computational Linguistics* special issue on semantic role labeling (Màrquez et al., 2008) includes other interesting papers on the topic, leveraging the PropBank conventions for labeling shallow semantic structures. Recently, there have been initiatives to predict syntactic dependencies as well as PropBank-style predicate-argument structures together using one joint model (Hajič et al., 2009).

In our work, we focus on the related topic of frame-semantic parsing. Note that from the annotated semantic roles for the two verbs in the sentence of Figure 2.1, it is unclear what the core roles ARG1 or ARG2 represent linguistically. To better understand the roles' meaning for a given verb, one has to refer to a verb specific file provided along with the PropBank corpus. Although collapsing these verb specific core roles into tags ARG0-ARG5 leads to a small set of classes to be learned from a reasonable sized corpus, analysis shows that the roles ARG2-ARG5 serve as many different roles for different verbs. Yi et al. (2007) point out that these four roles are highly overloaded and inconsistent, and they mapped them to VerbNet (Schuler, 2005) thematic roles to get improvements on the SRL task. Instead of working with PropBank, we focus on shallow semantic parsing of sentences in the paradigm of frame semantics (Fillmore, 1982). We discuss related work on frame-semantic parsing below.

2.2.2 Frame-Semantic Parsing

The FrameNet lexicon (Fillmore et al., 2003) contains rich linguistic information about lexical items and predicate-argument structures. A semantic frame present in this lexicon has associated words and phrases that can potentially evoke it in a natural language utterance. Each frame has associated roles, which are also enumerated in the lexicon. Figure 2.2 shows frame-semantic annotations for the same sentence shown in Figure 2.1. Note that the verbs "created" and "pushed" evoke the semantic frames INTENTIONALLY_CREATE and CAUSE_CHANGE_POSITION_ON_A_SCALE respectively. The corresponding *lexical units*, *create*.V and *push*.V (See §5.3.1 for a detailed description of lexical units.) from the FrameNet lexicon are also shown in the figure right above the semantic frames. The PropBank analysis in Figure 2.1 also had predicate-argument annotations for these two verbs. While PropBank labeled the roles of these verbs with its limited set of tags, the frame-semantic parse labels the frames' arguments with specific roles shown in the figure, making it immediately clear what those arguments mean. For example, for the INTENTIONALLY_CREATE frame, "more than 1.2 million jobs" is the Created_entity, and "In that time" is the Time when the jobs were created. FrameNet also allows non-verbal words and phrases to evoke semantic frames. As an example, the nominal "million" in the sentence evokes the frame CARDINAL_NUMBERS, and uses "jobs" as the Entity role, that is enumerated by the cardinal number, "1.2" serves as the argument filling the Multiplier role and "more than" satisfies the Precision role. FrameNet goes beyond other annotation projects like NomBank (Meyers et al., 2004) that focus on nouns in that it even allows adjectives, adverbs and prepositions to evoke frames. Finally, similar words and phrases are grouped together under a semantic frame in this lexicon and both frames and roles are organized in a hierarchy to provide itself a structure unlike PropBank, which does not relate words or phrases.

Most of early work on frame-semantic parsing has made use of the *exemplar* sentences in the FrameNet corpus (see §5.1.1), each of which is annotated for a single frame and its arguments. Gildea and Jurafsky (2002) presented a discriminative model for arguments given the frame; Thompson et al. (2003) used a generative model for both the frame and its arguments; and Fleischman et al. (2003) first used maximum entropy models to find and label arguments given the frame. Shi and Mihalcea (2004) developed a rule-based system to predict frames and their arguments in text, and Erk and Padó (2006) introduced the Shalmaneser tool, which employs Naïve Bayes classifiers to do the same. Other FrameNet SRL systems (Giuglea and Moschitti, 2006, for instance) have used SVMs. Most of this work was done on an older, smaller version of FrameNet, containing around 300 frames and less than 500 unique semantic roles. Unlike this body of work, we experimented with the larger SemEval 2007 shared task dataset, and also the newer FrameNet v. 1.5,[2] which lists 877 frames and 1068 role types, thus handling many more labels, and resulting in richer frame-semantic parses.

Recent work in frame-semantic parsing—in which sentences may contain multiple frames which need to be recognized along with their arguments—has been first undertaken during the SemEval'07 task 19 of frame-semantic structure extraction (Baker et al., 2007), and is a focus of this thesis. This task leveraged FrameNet v. 1.3, and also released a small corpus containing a little more than 2000 sentences with full text annotations. The LTH system of Johansson and Nugues (2007), which we use as our baseline (§5.1.4), had the best performance in the SemEval'07 task in terms of full frame-semantic parsing. Johansson and Nugues (2007) broke down the task as identifying targets that could evoke frames in a sentence, identifying the correct semantic frame for a target, and finally determining the arguments that fill the semantic roles of a frame. They used

[2]Available at `http://framenet.icsi.berkeley.edu` as of May 18, 2012.

a series of SVMs to classify the frames for a given target, associating unseen lexical items to frames and identifying and classifying token spans as various semantic roles. Both the full text annotation corpus as well as the FrameNet exemplar sentences were used to train their models. Unlike Johansson and Nugues, we use *only* the full text annotated sentences as training data and model the whole problem with only two statistical models. Our system results in significantly better overall parsing scores. We also model the argument identification problem (or semantic role labeling) using a joint structure prediction model; furthermore semi-supervised learning is used to improve predicate coverage. We also present experiments on recently released FrameNet 1.5 data.

Among other work based on FrameNet, Matsubayashi et al. (2009) investigated various uses of relations in the FrameNet taxonomy for learning generalizations over roles; they trained a log-linear model on the SemEval'07 data to evaluate features for the subtask of argument identification. Another line of work has sought to extend the coverage of FrameNet by exploiting VerbNet and WordNet (Shi and Mihalcea, 2005; Giuglea and Moschitti, 2006; Pennacchiotti et al., 2008), and projecting entries and annotations within and across languages (Boas, 2002; Fung and Chen, 2004; Padó and Lapata, 2005b; Fürstenau and Lapata, 2009b). Others have explored the application of frame-semantic structures to tasks such as information extraction (Moschitti et al., 2003; Surdeanu et al., 2003), textual entailment (Burchardt, 2006; Burchardt et al., 2009), question answering (Narayanan and Harabagiu, 2004; Shen and Lapata, 2007), and paraphrase recognition (Padó and Erk, 2005).

2.2.3 Semi-Supervised Methods

Although there has been a significant amount of work in supervised shallow semantic parsing using both PropBank- and FrameNet-style representations, improvements over vanilla supervised methods by using unlabeled data have not been very common. Fürstenau and Lapata (2009b) present a method of projecting predicate-argument structures from some seed examples to unlabeled sentences, and use a linear program formulation to find the best alignment explaining the projection. Next, the projected information as well as the seeds are used to train statistical model(s) for SRL. The authors ran experiments using a set of randomly chosen verbs from the exemplar sentences of FrameNet and found improvements over supervised methods. In an extension to this work, Fürstenau and Lapata (2009a) present a method for finding examples for unseen verbs using a graph alignment method; this method represents sentences and their syntactic analysis as graphs and graph alignment is used to project annotations from seed examples to unlabeled sentences. This alignment problem is again modeled as a linear program. Although this line of work presents a novel direction in the area of SRL, the work does not deal with non-verbal predicates, and does not evaluate the presented methods on the full-text annotations of the FrameNet release. Hence, it is difficult to measure its efficacy over state-of-the-art systems, such as the ones participating in the SemEval 2007 shared task.

Deschacht and Moens (2009) present a technique of incorporating additional information from unlabeled data by using a latent words language model. Latent variables are used to model the underlying representation of words, and parameters of this model are estimated using standard unsupervised methods. Next, the latent information is used as features for an SRL model. Improvements over supervised SRL techniques are observed with the augmentation of these extra features. The authors also compare their method with the aforementioned two methods of Fürstenau and Lapata (2009a,b) and show relative improvements. Experiments are performed on the CoNLL 2008 shared task dataset (Surdeanu et al., 2008), which follows the PropBank conventions and only labels verbal and nominal predicates in contrast to our work, where we focus on most syntactic categories. A similar approach is presented by Weston et al. (2008) who use neural embeddings of words, which are eventually used for SRL; improvements over state-of-the-art PropBank style SRL systems are observed.

In our work, we depart from the aforementioned related work in that we strive to improve the performance of our supervised frame-semantic parser on unseen predicates. We achieve this goal by using a graph-based semi-supervised learning approach that learns distributions over semantic frames potentially evoked by unknown predicates. These distributions are next used as constraints during inference; the statistical model used for inference is trained only on supervised data. We evaluate our model on standard FrameNet full text annotations, using all syntactic categories to which the FrameNet lexical units belong.

2.2.4 Comparison with Deep Semantic Parsing

In this subsection, we compare the formalism for shallow semantic analysis that we have adopted in this thesis with deeper semantic parsing formalisms. Automatically mapping natural language sentences to typed lambda calculus encodings has attracted interest in the NLP community (Zettlemoyer and Collins, 2005). Combinatory categorical grammar (Steedman, 1996, 2000) has been used in this work as the grammar formalism as it combines both syntax and semantics making use of a compositional semantics based on lambda calculus. Sophisticated learning techniques result in parsing models that are used to answer database queries in restricted domains. Zettlemoyer and Collins (2007) present an extension to this work, where a better learning strategy is adopted, and the queries are evaluated on a diverse dataset. Recent work in this strain of semantic parsing research has focused on learning factored lexicons by learning lexemes that pair words with logical constants and lexical templates, that map lexemes to full lexical items (Kwiatkowski et al., 2011). This sort of factored learning is intuitive and is analogous to semantic frames that appear commonly in the SRL literature.

A parallel line of work has focused on techniques that have not relied on lambda calculus expressions but dealt with other meaning representations such as CLANG, which is a formal language used for the RoboCup competitions.[3] Another example is the Geoquery formal language which resembles Prolog but also consists of meta-level predicates (Zelle and Mooney, 1996). Kate et al. (2005) present a method for inducing tranformation rules to transform natural language sentences to such logical forms. Ge and Mooney (2005) present a technique to use phrase-structure syntax trees and associate them with meaning representations, which are learned automatically. Statistical machine translation techniques have also been used to find logical forms for natural language sentences by using synchronous grammars and alignment techniques popular in the MT community (Wong and Mooney, 2006, 2007).

Very recently, there has been a focus on the learning logical forms using indirect supervision. Most often, the indirect set of supervision comes in the form of natural language questions or queries to a database, and the corresponding answer. Examples of this line of work are (Clarke et al., 2010) and (Liang et al., 2011). The latter provides a new dependency based semantic representation which is never explicitly presented to the system but is learned using indirect supervision in the form of answers gathered from relational databases. Liang et al. (2011) present experiments with best results on a question-answering task which previous work has dealt with; however, previous systems were trained using direct supervision in the form of logical structures paired with natural language sentences.

There are several advantages of the deep logical structures that the aforementioned work models. One of the more important advantages is the fact that these representations provide a global structure over a sentence which models the compositionality of the individual lexical items' meanings. Second, they also model logical operators like negation and quantification, which are essential for natural language inference. Shallower representations such as frame-semantic parses or PropBank-style predicate argument structures are unable to handle such operators; neither do they provide a global structure modeling every word of a sentence. However, most of the work involving deep semantics has focused on extremely narrow domains from which these structures are learned. These narrow domains lack syntactic and lexical-semantic diversity which is rampant in unrestricted text such as news, which other NLP applications focus on. Hence, learning robust deep semantic parsers that work out of the box still remains a big challenge. The line of work that relies on indirect supervision, e.g. the use of feedback from a relational database, works under the assumption that such databases are readily available for a domain, which is usually not the case. Although this research presents great promise, we are yet to see deep semantic parsers that can be used in other NLP applications.

[3]See http://www.robocup.com.

Although shallower in comparison to logical forms, frame-semantic parses produce richer structures than PropBank semantic roles, and model a larger set of predicates. In our work, we also provide semi-supervised extensions for the handling of unseen predicates, for further robust coverage. Several NLP applications have started using such predicate-argument structures because of their wide coverage, a recent example being modern question-answering systems (Ferrucci et al., 2008) that leverage automatic semantic role annotations for constraining information retrieval mechanisms that form these systems' core (Bilotti et al., 2007).

Chapter 3

Modeling Tools

Here, we describe a set of tools used across the two problems in computational semantics that we address in this thesis. These tools are general, and have been used in a wide variety of problems in natural language processing. Since they do not integrate into the two major semantic processing tasks we consider and because these tools appear frequently in the following few chapters, we carve out their description as subsections in this chapter. §3.1 describes briefly the formalism of dependency grammar, and provides the notation used for dependency trees in this work. §3.2 explains the basics of log-linear moels, mentions the use of latent-variables, and optimization methods used for training them. §3.3 focuses on how MapReduce, a distributed framework for computing, can be used for data and computation intensive tasks relating to these tools. §3.4 briefly looks at WordNet and how it is used for our problems.

3.1 Dependency Trees

A dependency tree is a lightweight syntactic representation. Given a sentence, a dependency tree assigns each word a syntactic parent, resulting in a graph with the words as its nodes, and the syntactic relationships as directed edges. An additional constraint ensures that the graph is a tree. In Chapter 1, we have already seen a dependency tree in Figure 1.2.

Figures 3.1 and 3.2 show two more dependency trees. The former is a *projective* dependency tree, where arcs cannot cross when they are depicted on one side of the sentence, while the latter is a *non-projective* tree where this constraint is not imposed. We have included a dummy root symbol “$” which serves as the parent to the main verb of a sentence. Since English is mostly projective, in all our experiments, we use an implementation of the Eisner algorithm (Eisner, 1996) available in the MST parser

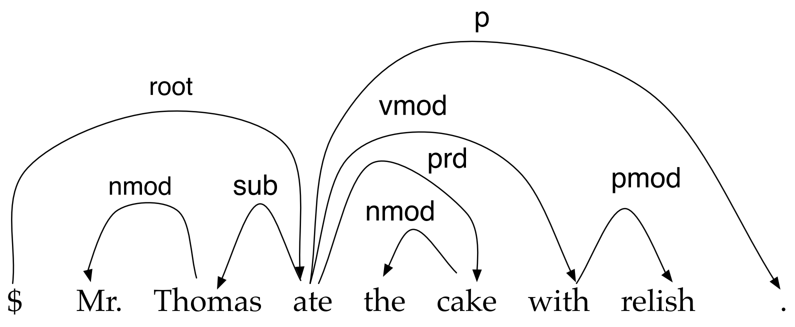


Figure 3.1: A projective dependency parse.

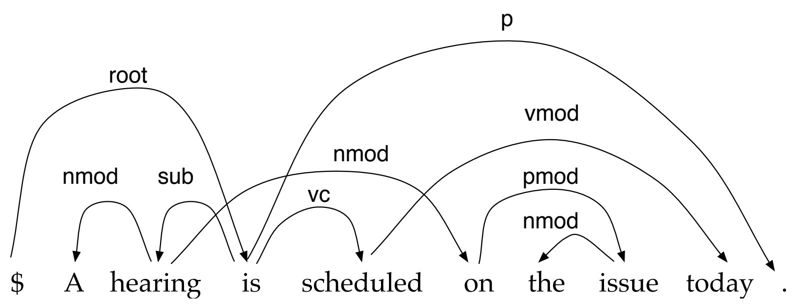


Figure 3.2: A non-projective dependency parse.

(McDonald et al., 2005). However, the publicly available version of the MST parser[1] performs parsing and the labeling of arcs jointly. We modified this to perform unlabeled parsing first, followed by the labeling of arcs using a log-linear classifier (Martins et al., 2008), which results in better labeled dependency accuracy. We trained it on sections 2–21 of the *WSJ* portion of the Penn Treebank, transformed to dependency trees following Yamada and Matsumoto (2003).

In the following chapters, we denote a dependency graph on a sentence $\mathbf{x} = \langle x_1, ..., x_k \rangle$ as $\tau^{\mathbf{x}}$. Because of the tree constraint, cycles are not allowed in this graph, and x_0 is taken to be the dummy "wall" symbol \$, whose only child is the root word of the sentence (normally the main verb). The tree consists of two mappings. The first is a mapping of word indices to indices of syntactic parents, $\tau_p : \{1, ..., k\} \rightarrow \{0, ..., k\}$. The second is a mapping of indices of words to dependency relation types in $\mathscr{L}$, the possible set of labels in the dependency grammar. It is defined as $\tau_l : \{1, ..., k\} \rightarrow \mathscr{L}$. The set of

[1]See `http://sourceforge.net/projects/mstparser`.

indices of x_i's children to its left is denoted by $\lambda^{\mathsf{x}}(i) = \{j : \tau^{\mathsf{x}}(j) = i, j < i\}$, and similarly the set of indices of children to its right is denoted by $\rho^{\mathsf{x}}(i) = \{j : \tau^{\mathsf{x}}(j) = i, j > i\}$. x_i has a single parent, denoted by $x_{\tau_p(i)}$. The label for x_i is denoted by $\tau_{\mathsf{l}}(i)$. Finally, the subtree rooted at the ith word by $\tau^{\mathsf{x},i}$.

3.2 Log-linear Models

Log-linear models (Berger, 1996) have been commonly used in natural language processing during the past two decades across a wide range of problems. A log-linear model defines a probability distribution over observation/label pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ as follows:

$$p_{\boldsymbol{\theta}}(x, y) = \frac{\exp \boldsymbol{\theta}^\top \mathbf{f}(x, y)}{\sum_{x', y'} \exp \boldsymbol{\theta}^\top \mathbf{f}(x', y')} \tag{3.1}$$

Equation 3.1 defines a joint distribution over the observations x and the labels y. Often, we prefer a conditional distribution, where we assume the observations as given, and we model the probability the labels:

$$p_{\boldsymbol{\theta}}(y \mid x) = \frac{\exp \boldsymbol{\theta}^\top \mathbf{f}(x, y)}{\sum_{y'} \exp \boldsymbol{\theta}^\top \mathbf{f}(x, y')} \tag{3.2}$$

The denominator in each of the two equations above is called the *partition function*. In the equations, $\mathbf{f} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$ denotes a feature vector, and $\boldsymbol{\theta} \in \mathbb{R}^d$ are model parameters estimated from data.

Given training data $\left\langle \langle x^{(i)}, y^{(i)} \rangle \right\rangle_{i=1}^{N}$, parameter estimation of a conditional model as in Equation 3.2 is frequently done using maximum a posteriori (MAP) estimation:

$$\boldsymbol{\theta}^* = \max_{\boldsymbol{\theta}} \; L(\boldsymbol{\theta}) - C\|\boldsymbol{\theta}\|_2^2 \tag{3.3}$$

where,

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(y^{(i)} \mid x^{(i)}) \tag{3.4}$$

In Equation 3.3, the term $C\|\boldsymbol{\theta}\|_2^2$, denotes a regularization term that prevents overfitting on the training data, and equates to a Gaussian prior distribution over the parameter space. This specific case is referred to as L_2 regularization as it takes the L_2 norm of the parameters, and other forms of regularization can be performed to get certain desired model properties. The hyperparameter C is often tuned over a development set or cross-validation is performed to get an optimal value.

In our work, the maximization procedure in Equation 3.3, is done using gradient-based methods. We employ a numerical batch optimization technique called L-BFGS (Liu and Nocedal, 1989) for a few problems, as well as a stochastic minibatch algorithm called stochastic gradient descent (Bottou, 2003). Both require us to compute the gradient of $L(\boldsymbol{\theta})$ with respect to the parameter vector. For the model expressed in Equation 3.2, the partial derivative of $L(\theta)$ with respect to one dimension θ_m of $\boldsymbol{\theta}$ is:

$$\frac{\partial L}{\partial \theta_m} = \sum_{i=1}^{N} \left(f_m(x^{(i)}, y^{(i)}) - \mathbb{E}_{p_{\boldsymbol{\theta}}(Y|x^{(i)})}[f_m(x^{(i)}, Y] \right) \tag{3.5}$$

In other words, it is the difference of the m^{th} feature's value in the data and the expected value of the same feature for all possible labels given an observation, under the conditional distribution. This kind of model estimation is also referred to as *discriminative training*.

Log-linear models are elegant probabilistic models in that they are capable of modeling overlapping features. Straightforward models like Equation 3.2 are also amenable to extensions with latent variables. For example, if we want to model unobserved latent variables z in our model, it can be expressed as:

$$\begin{aligned} p_{\boldsymbol{\theta}}(y \mid x) &= \sum_{z} p_{\boldsymbol{\theta}}(y, z \mid x) \\ &= \frac{\sum_{z} \exp \boldsymbol{\theta}^{\top} \mathbf{f}(x, y, z)}{\sum_{y', z'} \exp \boldsymbol{\theta}^{\top} \mathbf{f}(x, y', z')} \end{aligned} \tag{3.6}$$

Here, we are marginalizing out the unobserved latent variables z. Latent variable modeling is useful in many NLP problems. To cite an example, if x denotes a sentence, and we want to model the a phrase-structure tree y from the Penn Treebank, we may assume that there is a finer latent variable syntactic tree z, which is unobserved but can explain the sentence better. Petrov and Klein (2008) presented such a framework that resulted in better scores for the phrase-structure parsing task. While inference, such a model can be used to produce Viterbi labeling as a by-product to show interesting latent structure. This can be done as:

$$\langle \hat{y}, \hat{z} \rangle = \underset{y,z}{\operatorname{argmax}}\, p_{\boldsymbol{\theta}}(y, z \mid x) \tag{3.7}$$

We will investigate two probabilistic models in the following chapters that use latent variables to model unobserved phenomenon in supervised data. Notice that the parameter estimation procedure for a latent-variable log-linear model changes from Equation 3.3:

$$\boldsymbol{\theta}^* = \max_{\boldsymbol{\theta}} \underbrace{\sum_{i=1}^{N} \log \sum_{z} p_{\boldsymbol{\theta}}(y^{(i)}, z \mid x^{(i)})}_{L'(\boldsymbol{\theta})} - C\|\boldsymbol{\theta}\|_2^2 \tag{3.8}$$

The summation inside the $\log$ makes this function non-convex; in our work, irrespective of the possibility of settling on local minima, we use L-BFGS to train our latent-variable models. Previous work in natural language processing have used the same technique to train latent-variable models with success (Wang et al., 2007; Petrov and Klein, 2008). For this objective, The partial derivative form expressed in Equation 3.5 changes to:

$$\frac{\partial L'}{\partial \theta_m} = \sum_{i=1}^{N} \left(\mathbb{E}_{p_{\boldsymbol{\theta}}(y^{(i)}, Z \mid x^{(i)})}[f_m(x^{(i)}, y^i, Z] - \mathbb{E}_{p_{\boldsymbol{\theta}}(Y, Z \mid x^{(i)})}[f_m(x^{(i)}, Y, Z] \right) \tag{3.9}$$

Thus the derivative now is a difference of two expectation terms. Under the conditional distribution of y, z given x, the first term is the expected value of the m^{th} feature among all latent variables with the correct training label, and the second term is the expected value of the same feature among all latent variables and all labels.

3.3 Distributed Computing for Parameter Estimation

Often, gathering statistics such as derivatives for gradient based optimization or expected counts for algorithms such as Expectation-Maximization is a computationally expensive operation. In other cases, the total number of training examples is so large that gathering statistics for the entire dataset becomes expensive. Moreover, such optimization techniques are iterative and are run till convergence or a large number of iterations. Experimentation with different sets of features often becomes prohibitively slow for such large models. In our experiments, whenever we encounter either data-intensive or computation-intensive training tasks, we resort to parallelizing the optimization procedure using a large-scale computing framework called MapReduce (Dean and Ghemawat, 2008).

MapReduce has a very straightforward architecture. Data is provided to a MapReduce job in the form of key-value pairs. These key-value pairs are divided across a number of machines. Each map task receives a chunk of key-value pairs, and iterates over each one of them. Every key-value pair in one map task is processed to result in another form of key-value pair(s). These output key-value pairs from the map tasks are sorted and grouped on the basis of the keys. Next, these are divided among a number of reduce tasks. Each reduce task receives several keys, with each key associated with all its values. The reduce task then iterates over each key, and performs an operation with its values. This operation results in a final key-value pair. At the end of the pipeline, a set of unique keys with corresponding values are produced as output.

Every iteration of an optimization procedure can be easily parallelized using a MapReduce task. Figure 3.3 shows a prototypical example of how this can be done:

1. The set of training examples $\left\langle \langle x^{(i)}, y^{(i)} \rangle \right\rangle_{i=1}^{N}$ is divided among M map tasks, each getting a chunk of training examples, and the parameter vector $\boldsymbol{\theta}$.

2. Map task m processes the m^{th} chunk, and produces key-value pairs of the form $\left\langle \langle d, \partial_d^m \rangle \right\rangle_{d=1}^{D}$, where d is a dimension of the parameter vector, and ∂_d^m is a partial derivative of the log-likelihood of this data chunk, with respect to the d^{th} parameter. This partial can be substituted with expected counts if the procedure is the EM algorithm.

3. The partitioner/sorter sorts, groups and partitions these key-value pairs such that they are divided amongst R reducers, each getting a bunch of keys along with associated values of the form $\left\langle \langle d, \{\partial_d^1, \ldots \partial_d^M\} \rangle \right\rangle_{d=r_1}^{r_2}$.

4. Each reducer next sums up these partial derivatives and outputs the total partial derivative for the entire training set, of the form $\left\langle \langle d, \partial_d \rangle \right\rangle_{d=r_1}^{r_2}$. This summation can be substituted by the M-step for the EM algorithm.

5. A concatenation of the outputs from the reducers produces $\nabla_\theta L$, which is the derivative of the training set log-likelihood.

The above procedure can speed up training by several orders of magnitude, ease the experimentation process and enable swift feature engineering. We make use of the Hadoop architecture (White, 2009) as well as the MPI architecture (Gropp et al., 1994) to implement the above procedure for our experiments. Note that the MPI implementation does not use Step 3, the partitioning/sorting step for standard MapReduce implementations like Hadoop.[2]

To run L-BFGS (Liu and Nocedal, 1989) on such a parallel architecture (either Hadoop or MPI), we compute the partial derivatives as well as the objective function's value using the five steps enumerated above using several worker processes in a MapReduce fashion.[3] Once the objective and the partial derivatives are computed,

[2]See `http://hadoop.apache.org/`.

[3]Note that the objective function can be computed in a very straightforward fashion along with the partial derivatives, by using an extra key, with its value being the objective function's magnitude for a single chunk.

an L-BFGS step is taken in a master node and the new parameter values are computed. These new parameters are then used to compute the objective and the partial derivatives again in a distributed fashion. This iterative procedure continues until convergence or a fixed number of iterations.

3.4 WordNet

In our models, WordNet (Fellbaum, 1998) has been used as a lexical resource. We use the WordNet lexical database for the sole purpose of finding possible lexical and semantic relationships between two words, without considering their part-of-speech information. These relationships, often asymmetric, are enumerated below and are used as features in log-linear models:

1. IDENTICAL WORD: This relationship is self explanatory. It holds only when two words are identical.

2. SYNONYM: This relationship holds when two words are synonyms of each other according to the WordNet database. Example WordNet synonyms of each other are “death” and “demise.”

3. ANTONYM: This relationship holds when two words convey opposite meanings, e.g. “death” and “birth.”

4. HYPERNYM: A word’s hypernym is one which is more generic. For example, a hypernym of “clatter” is “noise.”

5. HYPONYM: Hyponym is the exact opposite of hypernym. A hyponym of “die” is “asphyxiate,” because to asphyxiate is more specific than to die.

6. DERIVED FORM: A derived form is a lexical relationship between two words and it holds if the second word is the lexical root of the first. For example, “probably” is a derived form of “probable.”

7. MORPHOLOGICAL VARIATION: A set of morphological variants of a verb consists of inflected forms of it, e.g. “passed” is a morphological variation of ”pass.”

8. VERB GROUP: The verb group relation is a symmetric relationship between two semantically related verbs. An example pair is “collaborate” and “cooperate.”

9. ENTAILMENT: The entailment relationship holds between two words if the first word implies the second word. For example, “snore” implies “sleep.”

10. SEE ALSO: This a semantic relationship between adjectives, which connects similar adjectives, but not exactly interchangeable ones, e.g. "different" and "incompatible."

11. CAUSAL RELATION: This is a relationship between two words when the second is caused by the first, e.g. "age" and "mature."

The tools presented in this chapter have been used pervasively in the following chapters. We use dependency trees produced by our implementation of the MST parser (McDonald et al., 2005; Martins et al., 2008) in the paraphrase identification problem (see §4.2.2) and for sentence preprocessing for the frame-semantic parsing problem (see §5.1.2) for deriving syntactic features. Log-linear models are used in the paraphrase problem (see §4.2.4) and for supervised modeling of the frame-semantic parsing problem (see §5.3.2 and §5.4.1). WordNet is also used in both problems – for the paraphrase problem, it is used to derive features for a lexical semantic transformation log-linear model (see §4.2.4) and in features for frame identification (see §5.3.2). Finally, parallel computation of statistics for training our models have been used in all our learning scenarios (see §4.2.6, §5.3.3, §5.4.2 and §6.2).

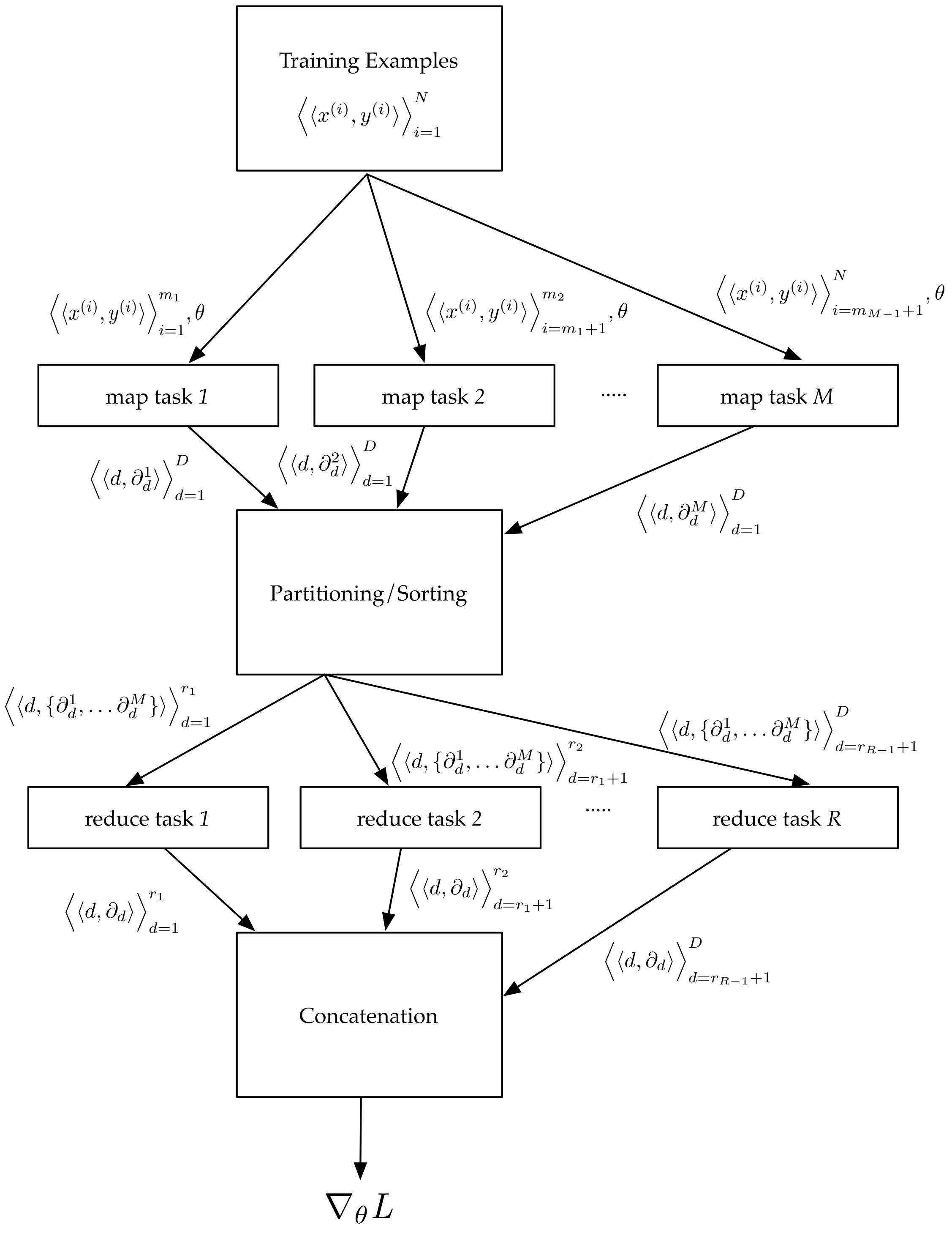

Training Examples
$\left\langle \langle x^{(i)}, y^{(i)} \rangle \right\rangle_{i=1}^{N}$
$\left\langle \langle x^{(i)}, y^{(i)} \rangle \right\rangle_{i=1}^{m_1}, \theta$
$\left\langle \langle x^{(i)}, y^{(i)} \rangle \right\rangle_{i=m_1+1}^{m_2}, \theta$
$\left\langle \langle x^{(i)}, y^{(i)} \rangle \right\rangle_{i=m_{M-1}+1}^{N}, \theta$
map task *1*
map task 2
.....
map task *M*
$\left\langle \langle d, \partial_d^1 \rangle \right\rangle_{d=1}^{D}$
$\left\langle \langle d, \partial_d^2 \rangle \right\rangle_{d=1}^{D}$
$\left\langle \langle d, \partial_d^M \rangle \right\rangle_{d=1}^{D}$
Partitioning/Sorting
$\left\langle \langle d, \{\partial_d^1, \ldots \partial_d^M\} \rangle \right\rangle_{d=1}^{r_1}$
$\left\langle \langle d, \{\partial_d^1, \ldots \partial_d^M\} \rangle \right\rangle_{d=r_1+1}^{r_2}$
$\left\langle \langle d, \{\partial_d^1, \ldots \partial_d^M\} \rangle \right\rangle_{d=r_{R-1}+1}^{D}$
reduce task *1*
reduce task 2
.....
reduce task *R*
$\left\langle \langle d, \partial_d \rangle \right\rangle_{d=1}^{r_1}$
$\left\langle \langle d, \partial_d \rangle \right\rangle_{d=r_1+1}^{r_2}$
$\left\langle \langle d, \partial_d \rangle \right\rangle_{d=r_{R-1}+1}^{D}$
Concatenation
$\nabla_\theta L$

Figure 3.3: Parallelizing one iteration of gradient-based optimization.

Chapter 4

Paraphrase Identification

In this chapter, we focus on the task of paraphrase identification, and present a novel method for recognizing paraphrases. The described work has been originally presented in Das and Smith (2009). The problem of modeling paraphrase relationships between natural language utterances has recently attracted interest. For computational linguists, solving this problem may shed light on how best to model the semantics of sentences. For natural language engineers, the problem bears on information management systems like abstractive summarizers that must measure semantic overlap between sentences (Barzilay and Lee, 2003), question answering modules (Marsi and Krahmer, 2005) and machine translation (Callison-Burch et al., 2006).

The paraphrase identification problem asks whether two sentences have essentially the same meaning. Although paraphrase identification is defined in semantic terms, it is usually solved using statistical classifiers based on shallow lexical, n-gram, and syntactic "overlap" features. Such overlap features give the best-published classification accuracy for the paraphrase identification task (Zhang and Patrick, 2005; Finch et al., 2005; Wan et al., 2006; Corley and Mihalcea, 2005, *inter alia*, see Chapter 2 for more details), but do not explicitly model correspondence structure (or "alignment") between the parts of two sentences. In our work, we adopt a model that posits correspondence between the words in the two sentences, defining it in *loose syntactic* terms: if two sentences are paraphrases, we expect their dependency trees to align closely, though some divergences are also expected, with some more likely than others. Following Smith and Eisner (2006), we adopt the view that the syntactic structure of sentences paraphrasing some sentence s should be "inspired" by the structure of s.

Because dependency syntax is still only a crude approximation to *semantic* structure, we augment the model with a lexical semantics component, based on WordNet, that models how *words* are probabilistically altered in generating a paraphrase. This combination of loose syntax and lexical semantics is similar to the "Jeopardy" model of Wang et al. (2007).

This syntactic framework represents a major departure from useful and popular surface similarity features, and the latter are difficult to incorporate into our probabilistic model. Therefore, we use a product of experts (Hinton, 2002) to bring together a logistic regression classifier built from n-gram overlap features and our syntactic model. This combined model leverages complementary strengths of the two approaches, outperforming a strong state-of-the-art baseline (Wan et al., 2006). We also compare our results with recently published work by Chang et al. (2010) and Socher et al. (2011) who model the same task and outperform our method by small margins.

The following sections are organized as follows. We introduce our probabilistic model in §4.1. The model makes use of three quasi-synchronous grammar models (Smith and Eisner, 2006, QG hereafter) as components (one modeling paraphrase, one modeling not-paraphrase, and one a base grammar); these are detailed, along with latent-variable inference and discriminative training algorithms, in §4.2. We discuss the Microsoft Research Paraphrase Corpus, upon which we conduct experiments, in §4.3. In §4.4, we present experiments on paraphrase identification with our model and make comparisons with the existing state-of-the-art. We describe the product of experts and our lexical overlap model, and discuss the results achieved in §4.5. Finally, we conclude with a discussion in §4.6.

4.1 Probabilistic Model

Since our task is a classification problem, we require our model to provide an estimate of the posterior probability of the relationship (i.e., "paraphrase," denoted p, or "not paraphrase," denoted n), given the pair of sentences.[1] Here, p_Q denotes model probabilities, c is a relationship class (p or n), and $\mathbf{s}_1$ and $\mathbf{s}_2$ are the two sentences. We choose the class according to:

$$\hat{c} \leftarrow \operatorname*{argmax}_{c \in \{\mathsf{p},\mathsf{n}\}} p_Q(c \mid \mathbf{s}_1, \mathbf{s}_2)$$

Using Bayes' rule, this can be written as:

$$\hat{c} \leftarrow \operatorname*{argmax}_{c \in \{\mathsf{p},\mathsf{n}\}} p_Q(c) \times p_Q(\mathbf{s}_1, \mathbf{s}_2 \mid c) \tag{4.1}$$

We define the class-conditional probabilities of the two sentences using the following generative story. First, grammar G_0 generates a sentence $\mathbf{s}$. Then a class c is chosen, corresponding to a class-specific *probabilistic quasi-synchronous grammar* G_c. (We will

[1] Although we do not explore the idea here, the model could be adapted for other sentence-pair relationships like entailment or contradiction.

discuss QG in detail in §4.2. For the present, consider it a specially-defined probabilistic model that generates sentences with a specific property, like "paraphrases s," when $c = \mathsf{p}$.) Given s, G_c generates the other sentence in the pair, s′.

When we observe a pair of sentences $\mathbf{s}_1$ and $\mathbf{s}_2$ we do not presume to know which came first (i.e., which was s and which was s′). Both orderings are assumed to be equally probable. For class c,

$$\begin{aligned} p_Q(\mathbf{s}_1, \mathbf{s}_2 \mid c) \quad = \quad & 0.5 \times p_Q(\mathbf{s}_1 \mid G_0) \times p_Q(\mathbf{s}_2 \mid G_c(\mathbf{s}_1)) + \\ & 0.5 \times p_Q(\mathbf{s}_2 \mid G_0) \times p_Q(\mathbf{s}_1 \mid G_c(\mathbf{s}_2)) \end{aligned} \tag{4.2}$$

where c can be p or n; $G_{\mathsf{p}}(\mathbf{s})$ is the QG that generates paraphrases for sentence s, while $G_{\mathsf{n}}(\mathbf{s})$ is the QG that generates sentences that are *not* paraphrases of sentence s. This latter model may seem counter-intuitive: since the vast majority of possible sentences are not paraphrases of s, why is a special grammar required? Our use of a G_{n} follows from the properties of the corpus currently used for learning, in which the negative examples were selected to have high lexical overlap. We return to this point in §4.3.

4.2 QG for Paraphrase Modeling

Here, we turn to the models G_{p} and G_{n} in detail.

4.2.1 Background

Smith and Eisner (2006) introduced the quasi-synchronous grammar formalism. Here, we describe some of its salient aspects. The model arose out of the empirical observation that translated sentences have *some* isomorphic syntactic structure, but divergences are possible. Therefore, rather than an isomorphic structure over a pair of source and target sentences, the syntactic tree over a target sentence is modeled by a source sentence-specific grammar "inspired" by the source sentence's tree. This is implemented by associating with each node in the target tree a subset of the nodes in the source tree. Since it loosely links the two sentences' syntactic structures, QG is well suited for problems like word alignment (Smith and Eisner, 2006), flexible translation models (Gimpel and Smith, 2009, 2011), parser projection (Smith and Eisner, 2009), and question answering (Wang et al., 2007).

Consider a very simple quasi-synchronous context-free dependency grammar that generates one dependent per production rule.[2] Let $\mathbf{s} = \langle s_1, ..., s_m \rangle$ be the source sen-

[2]Our actual model is more complicated; see §4.2.2.

tence. The grammar rules will take one of the two forms:

$$\langle t, l\rangle \rightarrow \langle t, l\rangle\langle t', k\rangle \quad \text{or}$$
$$\langle t, l\rangle \rightarrow \langle t', k\rangle\langle t, l\rangle$$

where t and t' range over the vocabulary of the target language, and l and $k \in \{0, ..., m\}$ are indices in the source sentence, with 0 denoting the null word \$.[3]

Hard or soft constraints can be applied between l and k in a rule. These constraints imply permissible "configurations." For example, requiring $l \neq 0$ and, if $k \neq 0$ then s_k must be a child of s_l in the source tree, we can implement a synchronous dependency grammar similar to that of Melamed (2004).

Smith and Eisner (2006) used a quasi-synchronous grammar to *discover* the correspondence between words implied by the correspondence between the trees. We follow Wang et al. (2007) in treating the correspondences as latent variables, and in using a WordNet-based lexical semantics model to generate the target words.

4.2.2 Detailed Model

We describe here how our novel approach models $p_Q(\mathbf{t} \mid G_\mathsf{p}(\mathbf{s}))$ and $p_Q(\mathbf{t} \mid G_\mathsf{n}(\mathbf{s}))$ for source and target sentences $\mathbf{s}$ and $\mathbf{t}$ (appearing in Equation 4.2 alternately as $\mathbf{s}_1$ and $\mathbf{s}_2$).

Consider two sentences: let the source sentence $\mathbf{s}$ contain m words and the target sentence $\mathbf{t}$ contain n words. Let the correspondence $a : \{1, ..., n\} \rightarrow \{0, ..., m\}$ be a mapping from indices of words in $\mathbf{t}$ to indices of words in $\mathbf{s}$. (We require each target word to map to at most one source word, though multiple target words can map to the same source word, i.e., $a(i) = a(j)$ while $i \neq j$.) When $a(i) = 0$, the ith target word maps to the wall symbol \$. Each of our QGs G_p and G_n generates the alignments a, the target dependency tree $\tau^{\mathbf{t}}$, and the sentence $\mathbf{t}$. Both G_p and G_n are structured in the same way, differing only in their parameters; henceforth we discuss G_p; G_n is similar.

We assume that the dependency parse trees of $\mathbf{s}$ and $\mathbf{t}$ are known.[4] Therefore our model defines:

$$\begin{aligned} p_Q(\mathbf{t} \mid G_\mathsf{p}(\mathbf{s})) &= p(\tau^{\mathbf{t}} \mid G_\mathsf{p}(\tau^{\mathbf{s}})) \\ &= \textstyle\sum_a p(\tau^{\mathbf{t}}, a \mid G_\mathsf{p}(\tau^{\mathbf{s}})) \end{aligned} \tag{4.3}$$

Because the QG is essentially a context-free dependency grammar, we can factor it into

[3] A more general QG could allow one-to-many alignments, replacing l and k with *sets* of indices.

[4] In our experiments, we use the MST parser as described in §3.1 to produce dependencies. Though this assumption of treating the parses as observed leads to a partial "pipeline" approximation of the posterior probability $p(c \mid \mathbf{s}, \mathbf{t})$, we believe that the relatively high quality of English dependency parsing makes this approximation reasonable.

recursive steps as follows (let i be an arbitrary index in $\{1, ..., n\}$):

$$\begin{aligned} P(\tau^{\mathbf{t},i} \mid t_i, a(i), \tau^{\mathbf{s}}) &= p_{val}(|\lambda^{\mathbf{t}}(i)|, |\rho^{\mathbf{t}}(i)| \mid t_i) \\ &\times \prod_{j \in \lambda^{\mathbf{t}}(i) \cup \rho^{\mathbf{t}}(i)} \sum_{a(j)=0}^{m} P(\tau^{\mathbf{t},j} \mid t_j, a(j), \tau^{\mathbf{s}}) \times p_{kid}(t_j, \tau_{\mathsf{l}}^{\mathbf{t}}(j), a(j) \mid t_i, a(i), \tau^{\mathbf{s}}) \end{aligned} \tag{4.4}$$

where p_{val} and p_{kid} are valence and child-production probabilities parameterized as discussed in §4.2.4. Note the recursion in the second-to-last line.

We next describe a dynamic programming solution for calculating $p(\tau^{\mathbf{t}} \mid G_{\mathsf{p}}(\tau^{\mathbf{s}}))$. In §4.2.4 we discuss the parameterization of the model.

4.2.3 Dynamic Programming

Let $C(i, l)$ refer to the probability of $\tau^{\mathbf{t},i}$, assuming that the parent of t_i, $t_{\tau_p^{\mathbf{t}}(i)}$, is aligned to s_l. For leaves of $\tau^{\mathbf{t}}$, the base case is:

$$C(i, l) = p_{val}(0, 0 \mid t_i) \times \sum_{k=0}^{m} p_{kid}(t_i, \tau_{\mathsf{l}}^{\mathbf{t}}(i), k \mid t_{\tau_p^{\mathbf{t}}(i)}, l, \tau^{\mathbf{s}})$$

where k ranges over possible values of $a(i)$, the source-tree node to which t_i is aligned. The recursive case is:

$$C(i, l) = p_{val}(|\lambda^{\mathbf{t}}(i)|, |\rho^{\mathbf{t}}(i)| \mid t_i) \times \sum_{k=0}^{m} p_{kid}(t_i, \tau_{\mathsf{l}}^{\mathbf{t}}(i), k \mid t_{\tau_p^{\mathbf{t}}(i)}, l, \tau^{\mathbf{s}}) \times \prod_{j \in \lambda^{\mathbf{t}}(i) \cup \rho^{\mathbf{t}}(i)} C(j, k) \tag{4.5}$$

We assume that the wall symbols t_0 and s_0 are aligned, so $p(\tau^{\mathbf{t}} \mid G_p(\tau^{\mathbf{s}})) = C(r, 0)$, where r is the index of the root word of the target tree $\tau^{\mathbf{t}}$. It is straightforward to show that this algorithm requires $O(m^2 n)$ runtime and $O(mn)$ space.

4.2.4 Parameterization

The valency distribution p_{val} in Equation 4.4 is estimated in our model using the dependency trees converted from the phrase-structure trees of the Penn Treebank, following Yamada and Matsumoto (2003). For unobserved cases, the conditional probability is estimated by backing off to the parent POS tag and child direction.

We discuss next how to parameterize the probability p_{kid} that appears in Equations 4.4, 4.5, and 4.5. This conditional distribution forms the core of our QGs, and we deviate from earlier research using QGs in defining p_{kid} in a fully generative way.

In addition to assuming that dependency parse trees for $\mathbf{s}$ and $\mathbf{t}$ are observable, we also assume each word w_i comes with POS and named entity tags. In our experiments these were obtained automatically using MXPOST (Ratnaparkhi, 1996) and BBN's Identifinder (Bikel et al., 1999).

For clarity, let $j = \tau_p^{\mathbf{t}}(i)$ and let $l = a(j)$.

$$\begin{aligned}
&p_{kid}(t_i, \tau_{\mathsf{l}}^{\mathbf{t}}(i), a(i) \mid t_j, l, \tau^{\mathbf{s}}) = \\
&p_{config}(config(t_i, t_j, s_{a(i)}, s_l) \mid t_j, l, \tau^{\mathbf{s}}) && (4.6)\\
&\quad \times p_{unif}(a(i) \mid config(t_i, t_j, s_{a(i)}, s_l)) && (4.7)\\
&\quad \times p_{lab}(\tau_{\mathsf{l}}^{\mathbf{t}}(i) \mid config(t_i, t_j, s_{a(i)}, s_l)) && (4.8)\\
&\quad \times p_{pos}(pos(t_i) \mid pos(s_{a(i)})) && (4.9)\\
&\quad \times p_{ne}(ne(t_i) \mid ne(s_{a(i)})) && (4.10)\\
&\quad \times p_{lsrel}(lsrel(t_i) \mid s_{a(i)}) && (4.11)\\
&\quad \times p_{word}(t_i \mid lsrel(t_i), s_{a(i)}) && (4.12)
\end{aligned}$$

We consider each of the factors above in turn.

Configuration In QG, "configurations" refer to the tree relationship among source-tree nodes (above, s_l and $s_{a(i)}$) aligned to a pair of parent-child target-tree nodes (above, t_j and t_i). In deriving $\tau^{\mathbf{t},j}$, the model first chooses the configuration that will hold among t_i, t_j, $s_{a(i)}$ (which has yet to be chosen), and s_l (line 4.6). This is defined for configuration c log-linearly by:[5]

$$p_{config}(\mathsf{c} \mid t_j, l, \tau^{\mathbf{s}}) = \frac{\alpha_{\mathsf{c}}}{\displaystyle\sum_{\mathsf{c}':\exists s_k, config(t_i, t_j, s_k, s_l) = \mathsf{c}'} \alpha_{\mathsf{c}'}} \tag{4.13}$$

Permissible configurations in our model are shown in Table 4.1. These are identical to prior work (Smith and Eisner, 2006; Wang et al., 2007), except that we add a "root" configuration that aligns the target parent-child pair to null and the head word of the source sentence, respectively. Using many permissible configurations helps remove negative effects from noisy parses, which our learner treats as evidence. Figure 4.1 shows some examples of major configurations that G_{p} discovers in the data.

Source tree alignment After choosing the configuration, the specific node in $\tau^{\mathbf{s}}$ that t_i will align to, $s_{a(i)}$ is drawn uniformly (line 4.7) from among those in the configuration

[5]We use log-linear models three times: for the configuration, the lexical semantics class, and the word. Each time, we are essentially assigning one weight per outcome and renormalizing among the subset of outcomes that are possible given what has been derived so far.

selected.

Dependency label, POS, and named entity class The newly generated target word's dependency label, POS, and named entity class are drawn from multinomial distributions p_{lab}, p_{pos}, and p_{ne} that condition, respectively, on the configuration and the POS and named entity class of the aligned source-tree word $s_{a(i)}$ (lines 4.8–4.10).

WordNet relation(s) The model next chooses a lexical semantics relation between $s_{a(i)}$ and the yet-to-be-chosen word t_i (line 4.11). Following Wang et al. (2007),[6] we employ a 13-feature log-linear model over all logically possible combinations of the 11 WordNet relations described in §3.4 as well as two more additional relations: whether the two words are same and is a number, and no relation. Similarly to Equation 4.13, we normalize this log-linear model based on the set of relations that are non-empty in WordNet for the word $s_{a(i)}$.

Word Finally, the target word is randomly chosen from among the set of words that bear the lexical semantic relationship just chosen (line 4.12). This distribution is, again, defined log-linearly:

$$p_{word}(t_i \mid lsrel(t_i) = \mathsf{R}, s_{a(i)}) = \frac{\alpha_{t_i}}{\sum_{w' : s_{a(i)} \mathsf{R} w'} \alpha_{w'}} \tag{4.14}$$

Here α_w is the Good-Turing unigram probability estimate of a word w from the Gigaword corpus (Graff, 2003).

4.2.5 Base Grammar G_0

In addition to the QG that generates a second sentence bearing the desired relationship (paraphrase or not) to the first sentence $\mathbf{s}$, our model in §4.1 also requires a base grammar G_0 over $\mathbf{s}$.

We view this grammar as a trivial special case of the same QG model already described. G_0 assumes the empty source sentence consists only of a single wall node. Thus every word generated under G_0 aligns to null, and we can simplify the dynamic programming algorithm that scores a tree $\tau^{\mathbf{s}}$ under G_0:

$$\begin{aligned} C'(i) = {} & p_{val}(|\lambda^{\mathbf{t}}(i)|, |\rho^{\mathbf{t}}(i)| \mid s_i) \\ & \times p_{lab}(\tau_{\mathfrak{l}}^{\mathbf{t}}(i)) \times p_{pos}(pos(t_i)) \times p_{ne}(ne(t_i)) \\ & \times p_{word}(t_i) \times \prod_{j:\tau^{\mathbf{t}}(j)=i} C'(j) \end{aligned} \tag{4.15}$$

[6]Note that Wang et al. (2007) designed p_{kid} as an interpolation between a log-linear lexical semantics model and a word model. Our approach is more fully generative.

Configuration	**Description**
parent-child	$\tau_p^{\mathbf{s}}(a(i)) = a(j)$, appended with $\tau_{\mathfrak{l}}^{\mathbf{s}}(a(i))$
child-parent	$a(i) = \tau_p^{\mathbf{s}}(a(j))$, appended with $\tau_{\mathfrak{l}}^{\mathbf{s}}(a(j))$
grandparent-grandchild	$\tau_p^{\mathbf{s}}(\tau_p^{\mathbf{s}}(a(i))) = a(j)$, appended with $\tau_{\mathfrak{l}}^{\mathbf{s}}(a(i))$
siblings	$\tau_p^{\mathbf{s}}(a(i)) = \tau_p^{\mathbf{s}}(a(j)), a(i) \neq a(j)$
same-node	$a(i) = a(j)$
c-command	the parent of one source-side word is an ancestor of the other source-side word
root	$a(j) = 0, a(i)$ is the root of s
child-null	$a(i) = 0$
parent-null	$a(j) = 0$, $a(i)$ is something other than root of s
other	catch-all for all other types of configurations, which are permitted

Table 4.1: Permissible configurations. i is an index in $\mathbf{t}$ whose configuration is to be chosen; $j = \tau_p^{\mathbf{t}}(i)$ is i's parent.

where the final product is 1 when t_i has no children. It should be clear that $p(\mathbf{s} \mid G_0) = C'(0)$.

We estimate the distributions over dependency labels, POS tags, and named entity classes using the transformed treebank (footnote 4). The distribution over words is taken from the Gigaword corpus (as in §4.2.4).

It is important to note that G_0 is designed to give a smoothed estimate of the probability of a particular parsed, named entity-tagged sentence. It is never used for parsing or for generation; it is only used as a component in the generative probability model presented in §4.1 (Equation 4.2).

4.2.6 Discriminative Training

Given training data $\left\langle \langle \mathbf{s}_1^{(i)}, \mathbf{s}_2^{(i)}, c^{(i)} \rangle \right\rangle_{i=1}^{N}$, we train the model discriminatively by maximizing regularized conditional likelihood (see §3.2). Let $\boldsymbol{\theta}$ represent the set of model parameters to be learned. $\boldsymbol{\theta}$ includes the class priors, the conditional distributions of the dependency labels given the various configurations, the POS tags given POS tags, the NE tags given NE tags appearing in expressions 4.8–4.10, the configuration weights

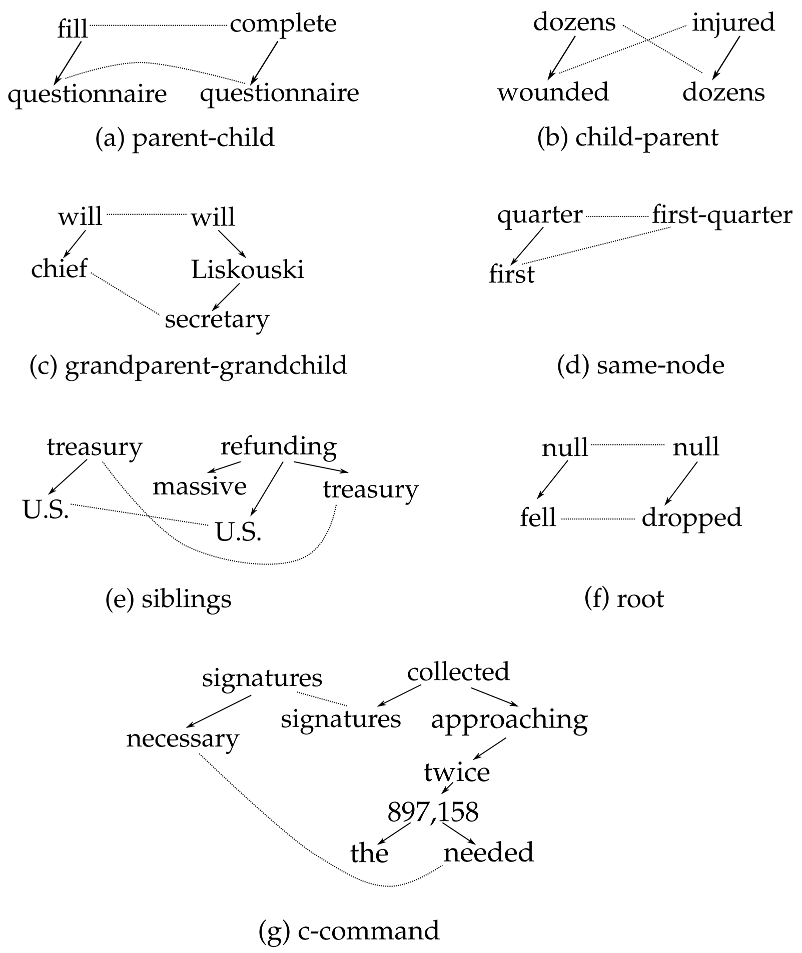


Figure 4.1: Some example configurations from Table 4.1 that G_{p} discovers in the dev. data. Directed arrows show head-modifier relationships, while undirected lines show alignments.

appearing in Equation 4.13, and the weights of the various features in the log-linear model for the lexical-semantics model. As noted, the distributions p_{val}, the word unigram weights in Equation 4.14, and the parameters of the base grammar are fixed using the treebank (see §4.2.4) and the Gigaword corpus.

Let the conditional probability of each training example's class, given the two sen-

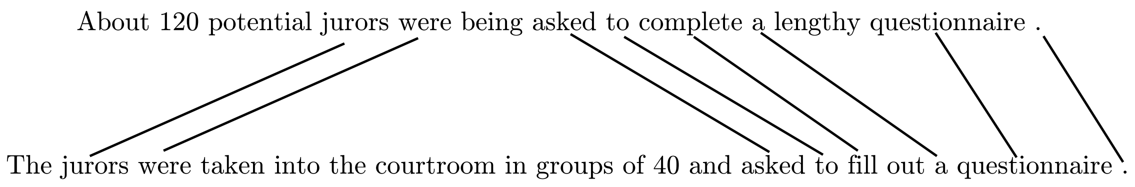

About 120 potential jurors were being asked to complete a lengthy questionnaire .
The jurors were taken into the courtroom in groups of 40 and asked to fill out a questionnaire .

Figure 4.2: Discovered alignment of Example 5.13 produced by G_{p}. Observe that the model aligns identical words and also "complete" and "fill" in this specific case. This kind of alignment provides an edge over a simple lexical overlap model.

tences be expressed by $p_Q(c^{(i)} \mid \mathbf{s}_1^{(i)}, \mathbf{s}_2^{(i)}, \boldsymbol{\theta})$. Note that Equation 4.2 relates this conditional probability to G_0, G_{p} and G_{n}. The discriminative training criterion maximizes the following criterion:

$$\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log p_Q(c^{(i)} \mid \mathbf{s}_1^{(i)}, \mathbf{s}_2^{(i)}, \boldsymbol{\theta}) - C\|\boldsymbol{\theta}\|_2^2 \tag{4.16}$$

Since there is a hidden variable (a), the objective function is non-convex (see §3.2). We locally optimize using the L-BFGS quasi-Newton method. Because many of our parameters are multinomial probabilities that are constrained to sum to one and L-BFGS is not designed to handle constraints, we treat these parameters as unnormalized weights that get renormalized (using a softmax function) before calculating the objective. Training is performed using MapReduce on 20 CPUs (see §3.3 for more details).

4.3 Data and Task

In our experiments, we have used the Microsoft Research Paraphrase Corpus (Dolan et al., 2004; Quirk et al., 2004). The corpus contains 5,801 pairs of sentences that have been marked as "equivalent" or "not equivalent." It was constructed from thousands of news sources on the web. Dolan and Brockett (2005) remark that this corpus was created semi-automatically by first training an SVM classifier on a disjoint annotated 10,000 sentence pair dataset and then applying the SVM on an unseen 49,375 sentence pair corpus, with its output scores skewed towards over-identification, i.e., towards generating some false paraphrases. 5,801 out of these 49,375 pairs were randomly selected and presented to human judges for refinement into true and false paraphrases. 3,900 of the pairs were marked as having "mostly bidirectional entailment," a standard definition of the paraphrase relation. Each sentence was labeled first by two judges, who averaged 83% agreement, and a third judge resolved conflicts. Note that this agreement percentage accounts for both true and false paraphrases.

We use the standard data split into 4,076 (2,753 paraphrase, 1,323 not) training and 1,725 (1147 paraphrase, 578 not) test pairs. We reserved a randomly selected 1,075 training pairs for tuning. We cite some examples from the training set here:

(4.17) Revenue in the first quarter of the year dropped 15 percent from the same period a year earlier.

With the scandal hanging over Stewart's company, revenue in the first quarter of the year dropped 15 percent from the same period a year earlier.

(4.18) About 120 potential jurors were being asked to complete a lengthy questionnaire.

The jurors were taken into the courtroom in groups of 40 and asked to fill out a questionnaire.

Example 4.17 is a true paraphrase pair. Notice the high lexical overlap between the two sentences (unigram overlap of 100% in one direction and 72% in the other). Example 4.18 is another true paraphrase pair with much lower lexical overlap (unigram overlap of 50% in one direction and 30% in the other). Notice the use of similar-meaning phrases and irrelevant modifiers that retain the same meaning in both sentences, which a lexical overlap model cannot capture easily, but a model like a QG might. Also, in both pairs, the relationship cannot be called total bidirectional equivalence because there is some extra information in one sentence which cannot be inferred from the other.

Example 4 was labeled "not paraphrase":

(4.19) "There were a number of bureaucratic and administrative missed signals - there's not one person who's responsible here," Gehman said.

In turning down the NIMA offer, Gehman said, "there were a number of bureaucratic and administrative missed signals here.

There is significant content overlap (unigram overlap of 65% in one direction and 100% in the other), making a decision difficult for a naïve lexical overlap classifier. (In fact, our model p_Q labels this example n while the lexical overlap models label it p.)

The fact that negative examples in this corpus were selected because of their high lexical overlap is important. It means that any discriminative model is expected to learn to distinguish mere overlap from paraphrase. This seems appropriate, but it does mean that the "not paraphrase" relation ought to be denoted "not paraphrase but deceptively similar on the surface." It is for this reason that we use a special QG for the n relation.

4.4 Experimental Evaluation

Here we present our experimental evaluation using p_Q. We trained on the training set (3,001 pairs) and tuned model metaparameters (C in Equation 4.16) and the effect of different feature sets on the development set (1,075 pairs). We report accuracy on the official MSRPC test dataset. If the posterior probability $p_Q(\mathtt{p} \mid \mathbf{s}_1, \mathbf{s}_2)$ is greater than 0.5, the pair is labeled "paraphrase" (as in Equation 4.1).

4.4.1 Comparison with Other Models

For comparison with other methods that have attempted to model the same task, first, we replicated a state-of-the-art baseline model for comparison. Wan et al. (2006) report a high accuracy model using a support vector machine. Thus, our baseline is a reimplementation of (Wan et al., 2006), using features calculated directly from $\mathbf{s}_1$ and $\mathbf{s}_2$ without recourse to any hidden structure: proportion of word unigram matches, proportion of lemmatized unigram matches, BLEU score (Papineni et al., 2001), BLEU score on lemmatized tokens, F measure (Turian et al., 2003), difference of sentence length, and proportion of dependency relation overlap. The SVM was trained to classify positive and negative examples of paraphrase using SVM^{light} (Joachims, 1999).[7] Metaparameters, tuned on the development data, were the regularization constant and the degree of the polynomial kernel (chosen in [10^{-5}, 10^2] and 1–5 respectively.). Our replication of the Wan et al. model is approximate, because we used different preprocessing tools: MXPOST for POS tagging (Ratnaparkhi, 1996), MST parser for parsing (McDonald et al., 2005, See §3.1), and Dan Bikel's interface (See http://www.cis.upenn.edu/~dbikel/software.html#wn) to WordNet (Fellbaum, 1998) for lemmatization information. Tuning led to $C = 17$ and polynomial degree 4.

It is unsurprising that the SVM performs very well on the MSRPC because of the corpus creation process (see Sec. 4.3) where an SVM was applied as well, with very similar features and a skewed decision process (Dolan and Brockett, 2005).

We also compare our results with the model of Chang et al. (2010), named as learning constrained latent representations (LCLR), who presented a method for paraphrase identification and also model latent alignments. Our final comparison is with the recursive auto-encoders (RAE) method of Socher et al. (2011). For both these methods, we make direct comparison with respective authors' reported results.

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

	Model	**Accuracy**	**Precision**	**Recall**
	all p	66.49	66.49	100.00
other models	Wan et al. (2006) – reported	75.63	77.00	90.00
	Wan et al. (2006) – replication	75.42	76.88	90.14
	Chang et al. (2010) – reported	76.41	-	-
	Socher et al. (2011) – reported	76.80	-	-
	lexical semantics features removed	68.64	68.84	96.51
p_Q	all features	73.33	74.48	91.10
	c-command disallowed (best; see text)	73.86	74.89	91.28
§4.5	p_L	75.36	78.12	87.44
	product of experts	76.06	79.57	86.05
	Wan et al. SVM and p_L	*80.17*	*100.00*	*92.07*
oracles	Wan et al. SVM and p_Q	*83.42*	*100.00*	*96.60*
	p_Q and p_L	*83.19*	*100.00*	*95.29*

Table 4.2: Accuracy, p-class precision, and p-class recall on the test set ($N = 1{,}725$). See text for differences in implementation between Wan et al. and our replication; their reported score does not include the full test set.

4.4.2 Results

Table 4.2 shows performance achieved by the baseline SVM and variations on p_Q on the test set. We performed a few feature ablation studies, evaluating on the development data. We removed the lexical semantics component of the QG,[8] and disallowed the syntactic configurations one by one, to investigate which components of p_Q contributes to system performance. The lexical semantics component is critical, as seen by the drop in accuracy from the table (without this component, p_Q behaves almost like the "all p" baseline). We found that the most important configurations are "parent-child," and "child-parent" while damage from ablating other configurations is relatively small. Most interestingly, disallowing the "c-command" configuration resulted in the best absolute accuracy, giving us the best version of p_Q. The c-command configuration allows more distant nodes in a source sentence to align to parent-child pairs in a target (see Figure 4.1d). Allowing this configuration guides the model in the wrong direction, thus reducing test accuracy. We tried disallowing more than one configuration at a time, without getting improvements on development data. We also tried ablating the WordNet relations, and observed that the "identical-word" feature hurt the model the most. Ablating the rest of the features did not produce considerable changes in accuracy.

The development data-selected p_Q achieves higher recall by 1 point than Wan et al.'s SVM, but has precision 2 points worse. We also come close to the LCLR method which does not model the problem as a monolingual translation scenario, but models hidden alignments using a constrained discriminative model.

4.4.3 Discussion

It is quite promising that a linguistically-motivated probabilistic model comes so close to a string-similarity baseline, *without* incorporating string-local phrases. We see several reasons to prefer the more intricate QG to the straightforward SVM. First, the QG discovers hidden alignments between words. Alignments have been leveraged in related tasks such as textual entailment (MacCartney and Manning, 2007, *inter alia*); they make the model more interpretable in analyzing system output (e.g., Figure 4.2). Second, the paraphrases of a sentence can be considered to be monolingual translations. We model the paraphrase problem using a direct machine translation model, thus providing a translation interpretation of the problem. This framework could be extended to permit paraphrase *generation*, or to exploit other linguistic annotations, such as representations of semantics (see, e.g., Qiu et al., 2006).

Nonetheless, the usefulness of surface overlap features is difficult to ignore. The

[8]This is accomplished by eliminating lines 4.11 and 4.12 from the definition of p_{kid} and redefining p_{word} to be the unigram word distribution estimated from the Gigaword corpus, as in G_0, without the help of WordNet.

RAE model of Socher et al. (2011) model (which achieves the best accuracy scores as shown in Table 4.2) both syntax and lexical overlap between the two given sentences in neural network model, and find benefits in terms of identification accuracy. Since our QG-based model cannot model lexical overlap directly, we next provide an efficient way to combine a surface model with p_Q.

4.5 Product of Experts

Incorporating structural alignment and surface overlap features inside a single model can make exact inference infeasible. As an example, consider features like n-gram overlap percentages that provide cues of content overlap between two sentences. One intuitive way of including these features in a QG could be including these only at the root of the target tree, i.e. while calculating $C(r, 0)$. These features have to be included in estimating p_{kid}, which has log-linear component models (Equation 4.6- 4.12). For these bigram or trigram overlap features, a similar log-linear model has to be normalized with a partition function, which considers the (unnormalized) scores of *all* possible target sentences, given the source sentence.

We therefore combine p_Q with a lexical overlap model that gives another posterior probability estimate $p_L(c \mid \mathbf{s}_1, \mathbf{s}_2)$ through a product of experts (PoE; Hinton, 2002):

$$p_J(c \mid \mathbf{s}_1, \mathbf{s}_2) = \frac{p_Q(c \mid \mathbf{s}_1, \mathbf{s}_2) \times p_L(c \mid \mathbf{s}_1, \mathbf{s}_2)}{\sum\limits_{c' \in \{\mathsf{p}, \mathsf{n}\}} p_Q(c' \mid \mathbf{s}_1, \mathbf{s}_2) \times p_L(c' \mid \mathbf{s}_1, \mathbf{s}_2)} \tag{4.20}$$

Equation 4.20 takes the product of the two models' posterior probabilities, then normalizes it to sum to one. PoE models are used to efficiently combine several expert models that individually constrain different dimensions in high-dimensional data, the product therefore constraining all of the dimensions. Combining models in this way grants to each expert component model the ability to "veto" a class by giving it low probability; the most probable class is the one that is least objectionable to all experts.

Probabilistic Lexical Overlap Model. We devised a logistic regression (LR) model incorporating 18 simple features, computed directly from $\mathbf{s}_1$ and $\mathbf{s}_2$, without modeling any hidden correspondence. LR provides a probability distribution (like the QG), but uses surface features (like the SVM). The features are of the form $precision_n$ (number of n-gram matches divided by the number of n-grams in $\mathbf{s}_1$), $recall_n$ (number of n-gram matches divided by the number of n-grams in $\mathbf{s}_2$) and F_n (harmonic mean of the previous two features), where $1 \leq n \leq 3$. We also used lemmatized versions of these features. This model gives the posterior probability $p_L(c \mid \mathbf{s}_1, \mathbf{s}_2)$, where $c \in \{\mathsf{p}, \mathsf{n}\}$. We estimated the model parameters analogously to Equation 4.16. Performance is reported in Table 4.2; this model is on par with the SVM, though trading recall in favor of precision. We view it as a probabilistic simulation of the SVM more suitable for combination with the QG.

Training the PoE Various ways of training a PoE exist. We first trained p_Q and p_L separately as described, then initialized the PoE with those parameters. We then continued training the parameters of both models jointly, maximizing (unregularized) conditional likelihood.

Experiment We used p_Q with the "c-command" configuration excluded, and the LR model in the product of experts. Table 4.2 includes the final results achieved by the PoE. The PoE model outperforms all the other models except the numbers reported by Chang et al. (2010) and Socher et al. (2011), achieving an accuracy of 76.06%.[9] The PoE is conservative, labeling a pair as p only if the LR and the QG give it strong p probabilities. This leads to high precision, at the expense of recall.

Oracle Ensembles Table 4.2 shows the results of three different oracle ensemble systems that correctly classify a pair if *either* of the two individual systems in the combination is correct. Note that the combinations involving p_Q achieve 83%, the human agreement level for the MSRPC. The LR and SVM are highly similar, and their oracle combination does not perform as well.

4.6 Conclusion

We have presented a probabilistic model of paraphrase incorporating syntax, lexical semantics, and hidden loose alignments between two sentences' trees. Though it fully defines a generative process for both sentences and their relationship, the model is discriminatively trained to maximize conditional likelihood. We showed that this model is competitive for determining whether there exists a semantic relationship between them, and can be improved by principled combination with a lexical overlap approach. Along with providing a posterior distribution over the class given two sentences, the quasi-synchronous grammars trained using our method can produce word level alignments between two sentences, if Viterbi inference is performed to obtain the latent correspondences. Figure 4.1 shows some example configurations discovered by the positive class QG G_{p}. Figure 4.2 shows a sentence pair with the Viterbi alignments decoded using the same QG.

[9]This accuracy is significant over p_Q under a paired t-test ($p < 0.04$), but is not significant over the SVM.

Chapter 5

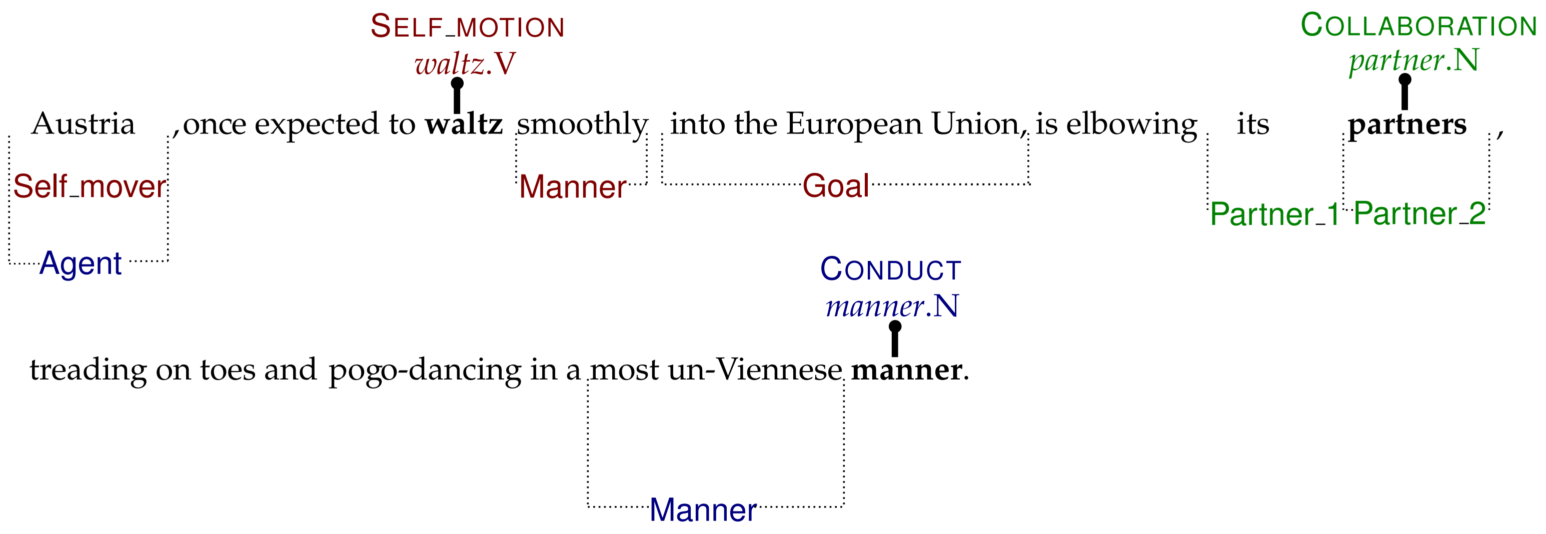
Frame-Semantic Parsing

FrameNet (Fillmore et al., 2003) is a linguistic resource storing considerable information about lexical and predicate-argument semantics in English. Grounded in the theory of frame semantics (Fillmore, 1982), it suggests—but does not formally define—a semantic representation that blends word-sense disambiguation and semantic role labeling.

In this chapter, we present a computational and statistical model for frame-semantic parsing, the problem of extracting from text semantic predicate-argument structures such as those shown in Figure 5.1. We aim to predict a frame-semantic representation as a *structure*, not as a pipeline of classifiers. We use a probabilistic framework that cleanly integrates the FrameNet lexicon and limited available training data. Although our models often involve strong independence assumptions, the probabilistic framework we adopt is highly amenable to future extension through new features, relaxed independence assumptions, and semi-supervised learning (as we observe in Chapter 6). Some novel aspects of our current approach include a latent-variable model that permits disambiguation of words not in the FrameNet lexicon, a unified model for finding and labeling arguments that diverges from prior work in semantic role labeling, and finally an exact dual decomposition algorithm that collectively predicts all the arguments of a frame together.

Our parser, named SEMAFOR[1] achieves the best published results to date on the SemEval'07 FrameNet task (Baker et al., 2007), and has been originally presented by Das et al. (2010). Herein, we present extensions of the aforementioned paper, pertaining to the set of features used, statistics of the datasets, and more error analysis. We also present results on newly released data with FrameNet 1.5, the newest edition of the lexicon, and a novel collective argument identification technique that makes use of an exact dual decomposition algorithm, earlier presented by Das et al. (2012).

[1]Semantic Analyzer of Frame Representations: http://www.ark.cs.cmu.edu/SEMAFOR



SELF_MOTION
waltz.V
COLLABORATION
partner.N
Austria, once expected to waltz smoothly into the European Union, is elbowing its partners,
Self_mover
Manner
Goal
Partner_1
Partner_2
Agent
CONDUCT
manner.N
treading on toes and pogo-dancing in a most un-Viennese manner.
Manner

Figure 5.1: An example sentence from the annotations released as part of FrameNet 1.5 with three targets marked in bold. Note that this annotation is partial because all potential targets have not been annotated with predicate-argument structures. Each target has its evoked semantic frame marked above it, in a distinct color. For each frame, its semantic roles are shown in the same color, and the spans fulfilling the roles are connected to the latter using dotted lines. For example, **manner** evokes the CONDUCT frame, and has the Agent and Manner roles fulfilled by "Austria" and "most un-Viennese" respectively.

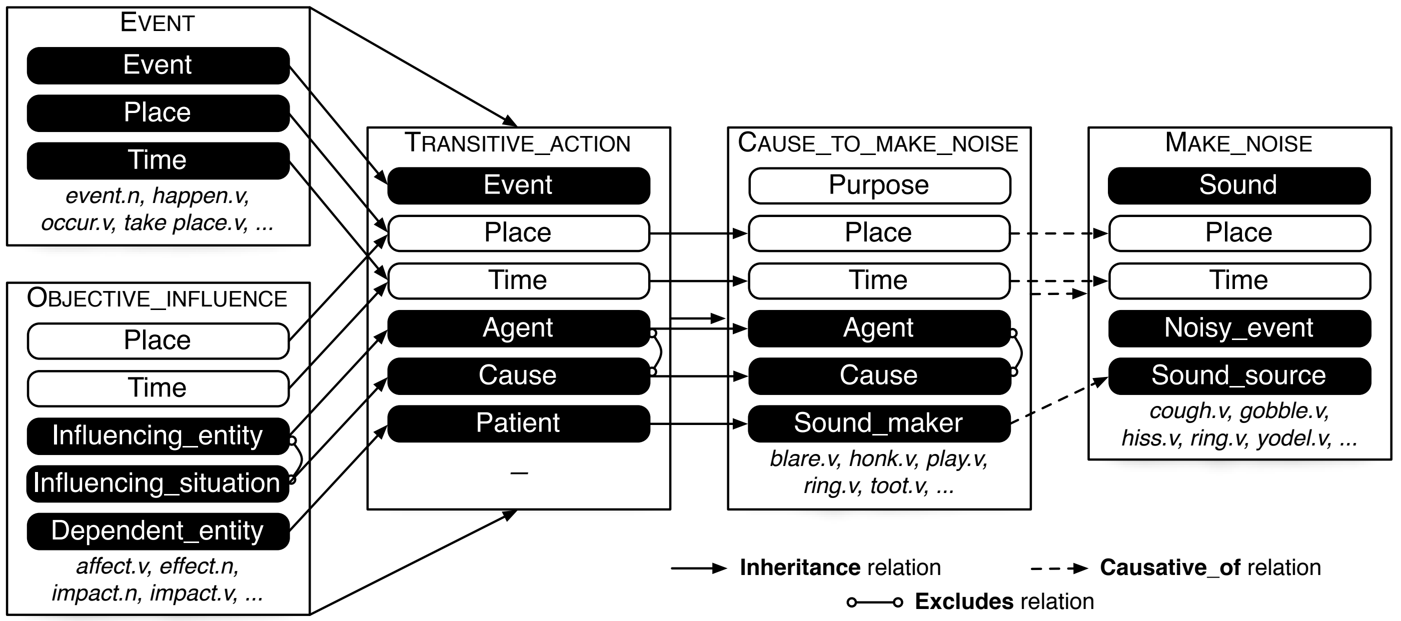


Figure 5.2: Partial illustration of frames, roles, and LUs related to the CAUSE_TO_MAKE_NOISE frame, from the FrameNet lexicon. "Core" roles are filled ovals. Non-core roles (such as Place and Time) as unfilled ovals. No particular significance is ascribed to the ordering of a frame's roles in its lexicon entry (the selection and ordering of roles above is for illustrative convenience). CAUSE_TO_MAKE_NOISE defines a total of 14 roles, many of them not shown here.

This chapter includes work done with Desai Chen and Nathan Schneider, who implemented parts of SEMAFOR in 2010, and André Martins, who implemented a general-purpose dual decomposition library that we apply for argument identification; the chapter is organized as follows. §5.1 describes in detail the task and the resources we used to solve it. §5.2 describes target identification, a heuristic technique to find salient targets in a sentence, §5.3 focuses on an automatic semantic frame disambiguation model, and §5.4 details the final subtask of argument identification; these are the three subproblems that constitute the full problem of frame-semantic parsing. In these sections, we also present the experiments and the results achieved on the SemEval'07 shared task for the individual components as well as the whole task, with comparisons with previous state of the art. §5.5, the penultimate section in this chapter, describes an alternate strategy of performing argument identification in comparison with §5.4 using a joint inference technique that incorporates several interesting linguistic constraints in a principled fashion; this section presents results achieved on the argument identification subtask on a newer version of the FrameNet data. Finally, §5.6 concludes this chapter with a discussion.

5.1 Resources and Task

Here, we consider frame-semantic parsing resources including datasets, evaluation strategies and previous baselines.

5.1.1 FrameNet Lexicon

The FrameNet lexicon is a taxonomy of manually identified general-purpose **frames** for English.[2] Listed in the lexicon with each frame are a set of lemmas (with parts of speech) that can denote the frame or some aspect of it—these are called **lexical units** (LUs). In a sentence, word or phrase tokens that evoke a frame are known as **targets**. The set of LUs listed for a frame in FrameNet may not be exhaustive; we may see a target in new data that does not correspond to an LU for the frame it evokes. Each frame definition also includes a set of frame elements, or **roles**, corresponding to different aspects of the concept represented by the frame, such as participants, props, and attributes. We use the term **argument** to refer to a sequence of word tokens annotated as filling a frame role. Figure 5.1 shows an example sentence from the training data with annotated targets, LUs, frames, and role-argument pairs. The FrameNet lexicon also provides information about relations between frames and between roles (e.g., INHERITANCE). Figure 5.2 shows a subset of the relations between three frames and their roles.

[2]Like the SemEval'07 participants, we used FrameNet v. 1.3 and also the newer version of the lexicon, namely FrameNet v. 1.5 (http://framenet.icsi.berkeley.edu).

	SemEval'07 Data	**FrameNet 1.5 Release**
	count	*count*
Number of exemplar sentences	139,439	154,607
Number of frame labels (types)	665	877
Number of role labels (types)	720	1,068
Number of sentences in training data	2,198	3,256
Number of targets in training data	11,195	19,582
Number of sentences in test data	120	2,420
Number of targets in test data	1,059	4,458

Table 5.1: Salient statistics of the datasets used in our experiments. There is a strong overlap between the two datasets.

Accompanying most frame definitions in the FrameNet lexicon is a set of lexicographic **exemplar sentences** (primarily from the British National Corpus) annotated for that frame. Typically chosen to illustrate variation in argument realization patterns for the frame in question, these sentences only contain annotations for a single frame.

In preliminary experiments, we found that using exemplar sentences directly to train our models hurt performance as evaluated on SemEval'07 data, which formed a benchmark for comparison with previous state of the art. This was a noteworthy observation, given that the number of exemplar sentences is an order of magnitude larger than the number of sentences in training data that we consider in our experiments (§5.1.2). This is presumably because the exemplars are neither representative as a sample nor similar to the test data. Instead, we make use of these exemplars in features (§5.3.2).

5.1.2 Data

In our experiments on frame-semantic parsing, we use two sets of data:

1. **SemEval'07 data:** In benchmark experiments for comparison with previous state of the art, we use a dataset that was released as part of the **SemEval 2007 shared task** on frame-semantic structure extraction. This dataset consisted of a few thousand sentences containing multiple targets, each annotated with a frame and their corresponding roles. The then current version of the lexicon, called **Framenet 1.3** was part of the shared task, and it provided with a list of frames, roles, and lexical units that could evoke a frame, along with rich ontological information such as the relationship between roles and frames, such as the ones shown in Figure 5.2.

Targets and Arguments by Part of Speech					
	targets			**arguments**	
	count	%		*count*	%
Noun	5155	52	Noun	9439	55
Verb	2785	28	Preposition or complementizer	2553	15
Adjective	1411	14	Adjective	1744	10
Preposition	296	3	Verb	1156	7
Adverb	103	1	Pronoun	736	4
Number	63	1	Adverb	373	2
Conjunction	8		Other	1047	6
Article	3				
	9824			17048	

Table 5.2: Breakdown of targets and arguments in the SemEval'07 training set in terms of part of speech. The target POS is based on the LU annotation for the frame instance. For arguments, this reflects the part of speech of the head word (estimated from automatic dependency parse); the percentage is out of all overt arguments.

The lexicon also contained 139,439 exemplar sentences containing one target each. The second column of Table 5.1 shows the statistics of the SemEval'07 data. The total number of frames, 665, indicate the frames we observed in the exemplars and the training portion of the data. The same holds for the number of role labels. We used the same training and test split as the SemEval'07 shared task; however, we took out four documents from the training set[3] for development. Table 5.2 shows some additional information about the SemEval dataset, indicating the syntactic categories of the targets and arguments in the data; the noticeable fact is the variety of syntactic categories that serve as targets, in contrast with the PropBank-style SRL task.

2. **Framenet 1.5 release:** A more recent version of the FrameNet lexicon was released in 2010.[4] We also test our statistical models (only frame identification and argument identification) on this dataset to get an estimate of how much improvement additional data can result in. Details of this dataset are shown in the third column of Table 5.1. We created a training and test split of the full text annotations

[3]StephanopoulousCrimes, Iran_Biological, NorthKorea_Introduction, and WMDNews_042106.

[4]Released on September 15, 2010, and downloadable from `http://framenet.icsi.berkeley.edu` as of May 18, 2012. In our experiments, we used a version downloaded on September 22, 2010.

released as part of FrameNet 1.5 ourselves; out of the 78 annotated documents released, we selected 55 for training containing 19,582 targets, while the remaining 23 containing 4,458 targets were set aside for testing. The number of target annotations per sentence in the test set were fewer than the training set for this dataset.[5] Appendix A gives the names of the test documents for fair replication of our work. We also randomly selected 4,462 targets from the training data for development, for the argument identification model (§5.4.1).

Preprocessing. We preprocess sentences in our dataset with a standard set of annotations: POS tags from MXPOST (Ratnaparkhi, 1996) and dependency parses from the MST parser as described in §3.1 since manual syntactic parses are not available for most of the FrameNet-annotated documents. We used WordNet (Fellbaum, 1998) for lemmatization. Our models treat these pieces of information as observations. We also labeled each verb in the data as having ACTIVE or PASSIVE voice, using code from the SRL system described by Johansson and Nugues (2008).

5.1.3 Task and Evaluation

Automatic annotations of frame-semantic structure can be broken into three parts: (1) *targets*, the words or phrases that evoke frames; (2) the *frame type*, defined in the lexicon, evoked by each target; and (3) the *arguments*, or spans of words that serve to fill roles defined by each evoked frame. These correspond to the three subtasks in our parser, each described and evaluated in turn: target identification (§5.2), frame identification (§5.3, not unlike word-sense disambiguation), and argument identification (§5.4, not unlike semantic role labeling).

The standard evaluation script from the SemEval'07 shared task calculates precision, recall, and F_1-measure for frames and arguments; it also provides a score that gives partial credit for hypothesizing a frame related to the correct one. We present precision, recall, and F_1-measure microaveraged across the test documents, report *labels-only* matching scores (spans must match exactly), and do not use named entity labels. More details can be found in the task description paper from SemEval 2007 (Baker et al., 2007) For our experiments, statistical significance is measured using a reimplementation of Dan Bikel's randomized parsing evaluation comparator,[6] a stratified shuffling test whose

[5]For creating the splits, we first included the documents that had incomplete annotations as mentioned in the initial FrameNet 1.5 data release in the test set; since we do not evaluate target identification for this version of data, the small number of targets per sentence does not matter. After these documents were put into the test set, we randomly selected 55 remaining documents for training, and picked the rest for additional testing. The final test set contains a total of 23 documents. As and when more annotations for the incomplete documents will be available, only testing needs to change, without any modification to training.

[6]See `http://www.cis.upenn.edu/~dbikel/software.html#comparator`.

original implementation is accompanied by the following description:

> The null hypothesis is that the two models that produced the observed results are the same, such that for each test instance [here, a set of predicate-argument structures for a sentence], the two observed scores are equally likely. This null hypothesis is tested by randomly shuffling individual sentences' scores between the two models and then re-computing the evaluation metrics [precision, recall or F_1 score in our case]. If the difference in a particular metric after a shuffling is equal to or greater than the original observed difference in that metric, then a counter for that metric is incremented. Ideally, one would perform all 2^n shuffles, where n is the number of test cases (sentences), but given that this is often prohibitively expensive, the default number of iterations is 10,000 [we use independently sampled 10,000 shuffles]. After all iterations, the likelihood of incorrectly rejecting the null [hypothesis, i.e., the p-value] is simply $(nc+1)/(nt+1)$, where nc is the number of random differences greater than the original observed difference, and nt is the total number of iterations.

Above, we quote the description from the aforementioned URL verbatim, with our explanations in square braces.

5.1.4 Baseline

A strong baseline for frame-semantic parsing is the system presented by (Johansson and Nugues, 2007, hereafter J&N'07), the best system in the SemEval'07 shared task. That system is based on a collection of SVMs. For frame identification, they used an SVM classifier to disambiguate frames for known frame-evoking words. They used WordNet synsets to extend the vocabulary of frame-evoking words to cover unknown words, and then used a collection of separate SVM classifiers—one for each frame—to predict a single evoked frame for each occurrence of a word in the extended set.

J&N'07 modeled the argument identification problem by dividing it into two tasks: first, they classified candidate spans as to whether they were arguments or not; then they assigned roles to those that were identified as arguments. Both phases used SVMs. Thus, their formulation of the problem involves a multitude of classifiers—whereas ours uses two log-linear models, each with a single set of weights, to find a full frame-semantic parse.

We compare our models with J&N'07 using the benchmark dataset from SemEval'07. However, since we are not aware of any other work using the FrameNet 1.5 fulltext annotations, we report our results on that dataset without comparison to any other system.

TARGET IDENTIFICATION	P	R	F_1
Our technique (§5.2)	**89.92**	**70.79**	**79.21**
Baseline: J&N'07	*87.87*	*67.11*	*76.10*

Table 5.3: Target identification results for our system and the baseline on the **SemEval'07 dataset**. Scores in bold denote significant improvements over the baseline ($p < 0.05$).

5.2 Target Identification

Target identification is the problem of deciding which word tokens (or word token sequences) evoke frames in a given sentence. In other semantic role labeling schemes (e.g., PropBank), simple part-of-speech criteria typically distinguish targets from non-targets. But in frame semantics, verbs, nouns, adjectives, and even prepositions can evoke frames under certain conditions. One complication is that semantically-impoverished **support predicates** (such as *make* in *make a request*) do not evoke frames in the context of a frame-evoking, syntactically-dependent noun (*request*). Furthermore, only temporal, locative, and directional senses of prepositions evoke frames.

Preliminary experiments using a statistical method for target identification gave us unsatisfactory results; instead, we followed J&N'07 in using a small set of rules to identify targets. First, we created a master list of all the morphological variants of targets that appear in the exemplar sentences and a given training set. For a sentence in new data, we considered only those substrings as candidate targets that appear in this master list. We also did not attempt to capture discontinuous frame targets: e.g. we treat *there would have been* as a single span even though the corresponding LU is *there be*.V.[7]

Next, we pruned the candidate target set by applying a series of rules identical to the ones described by (Johansson and Nugues, 2007, §3.1.1), with two exceptions. First, they identified locative, temporal, and directional prepositions using a dependency parser so as to retain them as valid LUs. In contrast, we pruned all types of prepositions because we found them to hurt our performance on the development set due to errors in syntactic parsing. In a second departure from their target extraction rules, we did not remove the candidate targets that had been tagged as support verbs for some other target.

Note that we used a conservative white list which filters out targets whose morphological variants were not seen either in the lexicon or the training data. Therefore, with this conservative process of automatically identifying targets, our *full* parser loses the capability to predict frames for completely unseen LUs, despite the fact that our our powerful frame identification model (§5.3) can accurately label frames for new LUs.

[7]There are 629 multiword LUs in the FrameNet 1.3 lexicon, corresponding to 4.8% of the targets in the

Results. Table 5.3 shows results on target identification tested on the SemEval'07 test set; our system gains 3 F_1 points over the baseline. This is statistically significant with $p < 0.01$. Our results are also significant in terms of precision ($p < 0.05$) and recall ($p < 0.01$). There are 85 distinct LUs for which the baseline fails to identify the correct target while our system succeeds. A considerable proportion of these units have more than one token (e.g. *chemical and biological weapon*.N, *ballistic missile*.N, etc.), which J&N'07 do not model. The baseline also does not label variants of *there be*.V, e.g. *there are* and *there has been*, which we correctly label as targets. Some examples of other single token LUs that the baseline fails to identify are names of months, LUs that belong to the ORIGIN frame (e.g. *iranian*.A) and directions, e.g., *north*.A or *north-south*.A.[8]

5.3 Frame Identification

Given targets, the parser next identifies their frames.

5.3.1 Lexical units

FrameNet specifies a great deal of structural information both within and among frames. For frame identification we make use of frame-evoking **lexical units**, the (lemmatized and POS-tagged) words and phrases listed in the lexicon as referring to specific frames. For example, listed with the BRAGGING frame are 10 LUs, including *boast*.N, *boast*.V, *boastful*.A, *brag*.V, and *braggart*.N. Of course, due to polysemy and homonymy, the same LU may be associated with multiple frames; for example, *gobble*.V is listed under both the INGESTION and MAKE_NOISE frames. We thus term *gobble*.V an **ambiguous** LU.[9] All targets in the exemplar sentences, and most in our training and test data, correspond to known LUs.

To incorporate frame-evoking expressions found in the training data but not the lexicon—and to avoid the possibility of lemmatization errors—our frame identification model will incorporate, via a latent variable, features based directly on exemplar and training **targets** rather than LUs. Let $\mathcal{L}$ be the set of (unlemmatized and automatically POS-tagged) targets found in the exemplar sentences of the lexicon and/or the sentences

SemEval 2007 training set; among them are *screw up*.V, *shoot the breeze*.V, and *weapon of mass destruction*.N. In the SemEval'07 training data, there are just 99 discontinuous multiword targets (1% of all targets).

[8] We do not evaluate the target identification module on the FrameNet 1.5 dataset; we just ran controlled experiments on that data to measure performance of the statistical frame identification and argument identification subtasks, assuming that the targets are given. Moreover the target annotations on the FrameNet 1.5 test set were fewer in number in comparison to the training set, resulting in a mismatch of target distributions between train and test settings.

[9] In our terminology an LU may be shared by multiple frames (LUs may be defined elsewhere as frame-specific).

- the POS of the parent of the head word of t_i
- the set of syntactic dependencies of the head word[11] of t_i
- if the head word of t_i is a verb, then the set of dependency labels of its children
- the dependency label on the edge connecting the head of t_i and its parent
- the sequence of words in the prototype, $\mathbf{w}_\ell$
- the lemmatized sequence of words in the prototype
- the lemmatized sequence of words in the prototype and their part-of-speech tags $\boldsymbol{\pi}_\ell$
- WordNet relation[12] ρ holds between ℓ and t_i
- WordNet relation[12] ρ holds between ℓ and t_i, and the prototype is ℓ
- WordNet relation[12] ρ holds between ℓ and t_i, the POS tag sequence of ℓ is $\boldsymbol{\pi}_\ell$, and the POS tag sequence of t_i is $\boldsymbol{\pi}_t$

Table 5.4: Features used for frame identification. All also incorporate f, the frame being scored. $\ell = \langle \mathbf{w}_\ell, \boldsymbol{\pi}_\ell \rangle$ consists of the words and POS tags[13] of a target seen in an exemplar or training sentence as evoking f.

in our training set. Let $\mathcal{L}_f \subseteq \mathcal{L}$ be the subset of these targets annotated as evoking a particular frame f.[10] Let $\mathcal{L}^l$ and $\mathcal{L}^l_f$ denote the lemmatized versions of $\mathcal{L}$ and $\mathcal{L}_f$, respectively. Then, we write *boasted*.VBD $\in \mathcal{L}_{\text{BRAGGING}}$ and *boast*.VBD $\in \mathcal{L}^l_{\text{BRAGGING}}$ to indicate that this inflected verb *boasted* and its lemma *boast* have been seen to evoke the BRAGGING frame. Significantly, however, another target, such as *toot your own horn*, might be used in other data to evoke this frame. We thus face the additional hurdle of predicting frames for unknown words.

The FrameNet annotators created new frames not present in the lexicon when they annotated the full text annotations, both for the SemEval'07 dataset as well as the FrameNet 1.5 release. We considered the union of the frames present in the exemplars (only the lexicon frames) and the frames in the training portions of our datasets; for SemEval'07, we observed 665 such frames and for the FrameNet 1.5 dataset, there were 877 frames. Automatically predicting new frames is a challenge not yet attempted to our knowledge (including here). Note that the scoring metric (§5.1.3) gives partial credit for *related* frames (e.g., a more general frame from the lexicon).

[10] For example, on average, there are 34 targets per frame in the SemEval'07 dataset; the average frame ambiguity of each target in $\mathcal{L}$ is 1.17.

[11] If the target is not a subtree in the parse, we consider the words that have parents outside the span, and apply three heuristic rules to select the head: 1) choose the first word if it is a verb; 2) choose the last word

5.3.2 Model

For a given sentence $\mathbf{x}$ with frame-evoking targets $\mathbf{t}$, let t_i denote the ith target (a word sequence).[14] Let t_i^l denote its lemma. We seek a list $\mathbf{f} = \langle f_1, \ldots, f_m \rangle$ of frames, one per target. In our model, the set of candidate frames for t_i is defined to include every frame f such that $t_i^l \in \mathcal{L}_f^l$—or if $t_i^l \not\in \mathcal{L}^l$, then every known frame (the latter condition applies for 4.7% of the gold targets in the SemEval 2007 development set). In both cases, we let $\mathcal{F}_i$ be the set of candidate frames for the ith target in $\mathbf{x}$. Also, let us denote the entire set of frames in the lexicon as $\mathcal{F}$.

To allow frame identification for targets whose lemmas were seen in neither the exemplars nor the training data, our model includes an additional variable, ℓ_i. This variable ranges over the seen targets in $\mathcal{L}_{f_i}$, which can be thought of as **prototypes** for the expression of the frame. Importantly, frames are *predicted*, but prototypes are summed over via the latent variable. The prediction rule requires a probabilistic model over frames for a target:

$$f_i \leftarrow \operatorname*{argmax}_{f \in \mathcal{F}_i} \sum_{\ell \in \mathcal{L}_f} p_{\boldsymbol{\theta}}(f, \ell \mid t_i, \mathbf{x}) \tag{5.1}$$

We model the probability of a frame f and the prototype unit ℓ, given the target and the sentence $\mathbf{x}$ as:

$$p_{\boldsymbol{\theta}}(f, \ell \mid t_i, \mathbf{x}) = \frac{\exp \boldsymbol{\theta}^\top \mathbf{g}(f, \ell, t_i, \mathbf{x})}{\sum_{f' \in \mathcal{F}} \sum_{\ell' \in \mathcal{L}_{f'}} \exp \boldsymbol{\theta}^\top \mathbf{g}(f', \ell', t_i, \mathbf{x})} \tag{5.2}$$

The above is a conditional log-linear model: for $f \in \mathcal{F}_i$ and $\ell \in \mathcal{L}_f$, where $\boldsymbol{\theta}$ are the model weights, and $\mathbf{g}$ is a vector-valued feature function. This discriminative formulation is very flexible, allowing for a variety of (possibly overlapping) features; e.g., a feature might relate a frame type to a prototype, represent a lexical-semantic relationship between a prototype and a target, or encode part of the syntax of the sentence.

Previous work has exploited WordNet for better coverage during frame identification (Johansson and Nugues, 2007; Burchardt et al., 2005, e.g., by expanding the set of targets using synsets), and others have sought to extend the lexicon itself (see §2.2). We differ in our use of a latent variable to incorporate lexical-semantic *features* in a discriminative model, relating known lexical units to unknown words that may evoke frames.

if the first word is an adjective; 3) if the target contains the word *of*, and the first word is a noun, we choose it. If none of these hold, choose the last word with an external parent to be the head.

[12]These are the 11 WordNet relations enumerated in §3.4 as well as NO RELATION.

[13]POS tags are found automatically during preprocessing.

[14]Each t_i is a word sequence $\langle x_u, \ldots, x_v \rangle$, $1 \leq u \leq v \leq n$, though in principle targets can be noncontiguous.

Here we are able to take advantage of the large inventory of partially-annotated exemplar sentences.

Note that this model makes a strong independence assumption: each frame is predicted independently of all others in the document. In this way the model is similar to J&N'07. However, ours is a single conditional model that shares features and weights across all targets, frames, and prototypes, whereas the approach of J&N'07 consists of many separately trained models. Moreover, our model is unique in that it uses a latent variable to smooth over frames for unknown or ambiguous LUs.

Frame identification features depend on the preprocessed sentence $\mathbf{x}$, the prototype ℓ and its WordNet lexical-semantic relationship with the target t_i, and of course the frame f. Our model uses binary features, which are detailed in Table 5.4.

5.3.3 Training

Given a training dataset (either SemEval'07 dataset or the FrameNet 1.5 full-text annotations), which is of the form $\langle\langle \mathbf{x}^{(j)}, \mathbf{t}^{(j)}, \mathbf{f}^{(j)}, \mathcal{A}^{(j)}\rangle\rangle_{j=1}^{N}$, we discriminatively train the frame identification model by maximizing the following log-likelihood:[15]

$$\max_{\boldsymbol{\theta}} \sum_{j=1}^{N} \sum_{i=1}^{m_j} \log \sum_{\ell \in \mathcal{L}_{f_i^{(j)}}} p_{\boldsymbol{\theta}}(f_i^{(j)}, \ell \mid t_i^{(j)}, \mathbf{x}^{(j)}) \tag{5.3}$$

Note that the training problem is non-convex because of the summed-out prototype latent variable ℓ for each frame. To calculate the objective function, we need to cope with a sum over frames and prototypes for each target (see Equation 5.2), often an expensive operation. We locally optimize the function using a distributed implementation of L-BFGS; although this learning algorithm cannot avoid local minima, prior work (Petrov and Klein, 2008, *inter alia*) have used this gradient-based method to optimize conditional log-likelihood of latent-variable models resulting in accurate systems. Our paraphrase identification model described in Chapter 4 had local minima as well, and we used L-BFGS to optimize the corresponding objective function. This is the most expensive model that we train: with 100 computers parallelized using the Hadoop implementation of MapReduce (see §3.3), training takes several hours. (Decoding takes only a few minutes on one CPU for the test set.) Each CPU of this cluster had 2 quad-core 1.86GHz CPUs with a total of 6GB of RAM each.[16]

[15]We found no benefit on either development dataset from using an L_2 regularizer (zero-mean Gaussian prior).

[16]We also used another implementation with 128 parallel cores in a multi-core MPI setup (Gropp et al., 1994). We resorted to this platform when the Hadoop cluster was no longer available to us. Each core of this setup had a clock rate of 2.27GHz and 8GB of RAM.

FRAME IDENTIFICATION (§5.3.2)		*targets*	exact frame matching *P*	exact frame matching *R*	exact frame matching F_1	partial frame matching *P*	partial frame matching *R*	partial frame matching F_1
SemEval'07 Data	Frame identification (oracle targets)	gold	60.21	60.21	60.21	74.21	74.21	74.21
	Frame identification (predicted targets)	auto §5.2	**69.75**	**54.91**	**61.44**	**77.51**	**61.03**	**68.29**
	Frame identification (J&N'07 targets)	auto	65.34	49.91	56.59	74.30	56.74	64.34
	Baseline: J&N'07	*auto*	*66.22*	*50.57*	*57.34*	*73.86*	*56.41*	*63.97*
FrameNet 1.5 Release	Frame identification (oracle targets)	gold	82.97	82.97	82.97	90.51	90.51	90.51

Table 5.5: Frame identification results on both the SemEval'07 dataset and the FrameNet 1.5 release. Precision, recall, and F_1 were evaluated under exact and partial frame matching; see §5.1.3. Bold indicates statistically significant results with respect to the baseline ($p < 0.05$).

5.3.4 Results

SemEval'07 Data. On the SemEval'07 dataset, We evaluate the performance of our frame identification model given gold-standard targets and automatically identified targets (§5.2); see Table 5.5. Given gold-standard targets, our model is able to predict frames for lemmas not seen in training, of which there are 210. The partial-match evaluation gives our model some credit for 190 of these, 4 of which are exactly correct. The hidden variable model, then, is finding related (but rarely exact) frames for unknown target words. The net effect of our conservative target identifier on F_1 is actually positive: the frame identifier is far more precise for targets seen explicitly in training.

Together, our target and frame identification outperform the baseline by 4 F_1 points. To compare the frame identification stage in isolation with that of J&N'07, we ran our frame identification model with the targets identified by their system as input. With partial matching, our model achieves a relative improvement of 0.6% F_1 over J&N'07, as shown in the third row of Table 5.5 (though this is not significant).

While our frame identification model thus performs on par with the current state of the art for this task, it improves upon J&N's formulation of the problem because it requires only a single model, learns lexical-semantic features as part of that model rather than requiring a preprocessing step to expand the vocabulary of frame-evoking words, and is probabilistic, which can facilitate global reasoning.

In the SemEval'07 dataset, for gold-standard targets, 210 out of 1058 lemmas were not present in the white list that we used for target identification (see §5.2). Our model correctly identifies the frames for 4 of these 210 lemmas. For 44 of these lemmas, the evaluation script assigns a score of 0.5 or more, suggesting that our model predicts a closely related frame. Finally, for 190 of the 210 lemmas, a positive score is assigned by the evaluation script. This suggests that the hidden variable model helps in identifying related (but rarely exact) frames for unseen targets, and explains why under exact—but not partial—frame matching, the F_1 score using automatic targets is commensurate with the score for oracle targets.[17]

For automatically identified targets, the F_1 score falls below 70 points because the model fails to predict frames for unseen lemmas. However, our model outperforms J&N'07 by 4 F_1 points. We measured statistical significance with respect the baseline for results with the partial frame matching criterion. The F_1 score of our model represents a significant improvement over the baseline ($p < 0.01$). The precision and recall measures are significant as well ($p < 0.05$ and $p < 0.01$, respectively). However, because targets

[17]J&N'07 did not report frame identification results for oracle targets; thus directly comparing the frame identification models is difficult. Considering only the predicted arguments for the frames they predicted correctly, we can estimate that their argument identification model given oracle targets and frames would have achieved 0.58 precision, 0.48 recall, and 0.53 F_1—though we caution that these are not directly comparable with our oracle results.

identified by J&N'07 and frames classified by our frame identification model resulted in scores on par with the baseline, we note that the significant results follow due to better target identification. Note from the results that the automatic target identification model show an increase in precision, at the expense of recall. This is because of the fact that the white list for target identification restricts the model to predict frames only for known LUs, leading to a more precise model.

FrameNet 1.5 Release. The last row of Table 5.5 shows results on the full text annotation test set of the FrameNet 1.5 release. Since the number of annotations nearly doubled, we see large improvements in frame identification accuracy. Note that we only evaluate the set up where gold targets were presented to the frame identifier. (As mentioned in §5.1.2, some documents in the test set has less number of targets per sentence, and auto target identification would overpredict targets for those sentences.)

Further experiments on this dataset were conducted to test the importance of the latent variable in our frame identification model. Recall that the decoding objective to choose the best frame f marginalizes over a latent variable ℓ, whose values range over targets known to be associated with f (see Equations 5.1–5.2). An alternative to having ℓ as a latent variable is to simply let it be equal to the target in consideration, without any marginalization, regardless of the frame under consideration; therefore, the features relating the prototype variable ℓ (see Table 5.4) are extracted for all 4,194 unique targets that were observed in training. Because each of these features needs to be associated with all 877 frames in the partition function, the result is an eighty-fold blowup of the feature space (the latent variable model had 465,317 features).

As this expansive feature set was beyond the scope of our engineering framework, we established a comparison excluding from the model all *unsupported features*, i.e. those features which would fire only in the partition function (never in the numerator). The model without a latent variable then becomes tractable, as it has 72,058 supported features. When trained and tested on the FrameNet 1.5 dataset, this model achieves an exact matching accuracy of 75.54% and a partial matching accuracy of 85.92%. If unsupported features are similarly excluded from the latent variable model, 165,200 features remain, and this model obtains an exact matching accuracy of 80.30% and a partial matching accuracy of 88.91%. Though slightly worse than the full latent variable model, this result is well above the comparable model without the latent variable. This establishes that the latent variable in our frame identification model helps in terms of accuracy, and lets us use a moderately sized feature set incorporating helpful unsupported features.

Finally, in our test set, we found that 144 out of the 4,458 annotated targets were unseen, and our full frame identification model only got 23.1% of the frames correct for those unseen targets; in terms of partial match accuracy, the model got a score of 46.6%. This, along with the results on the SemEval 2007 unseen targets, shows that there is substantial opportunity for improvement when unseen targets are presented to the system. We address this issue in Chapter 6.

5.4 Argument Identification

Given a sentence $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$, the set of targets $\mathbf{t} = \langle t_1, \ldots, t_m \rangle$, and a list of evoked frames $\mathbf{f} = \langle f_1, \ldots, f_m \rangle$ corresponding to each target, argument identification is the task of choosing which of each f_i's roles are filled, and by which parts of $\mathbf{x}$. This task is most similar to the problem of semantic role labeling, but uses frame-specific labels that are richer than the PropBank annotations.

5.4.1 Model

Let $\mathcal{R}_{f_i} = \{r_1, \ldots, r_{|\mathcal{R}_{f_i}|}\}$ denote frame f_i's **roles** (named frame element types) observed in an exemplar sentence and/or our training set. A subset of each frame's roles are marked as **core** roles; these roles are conceptually and/or syntactically necessary for any given use of the frame, though they need not be overt in every sentence involving the frame. These are roughly analogous to the core arguments ARG0–ARG5 in PropBank. Non-core roles—analogous to the various ARGM-* in PropBank—loosely correspond to syntactic adjuncts, and carry broadly-applicable information such as the time, place, or purpose of an event. The lexicon imposes some additional structure on roles, including relations to other roles in the same or related frames, and semantic types with respect to a small ontology (marking, for instance, that the entity filling the protagonist role must be sentient for frames of cognition). Figure 5.2 illustrates some of the structural elements comprising the frame lexicon by considering the CAUSE_TO_MAKE_NOISE frame.

We identify a set $\mathcal{S}$ of spans that are candidates for filling any role $r \in \mathcal{R}_{f_i}$. In principle, $\mathcal{S}$ could contain any subsequence of $\mathbf{x}$, but in this work we only consider the set of contiguous spans that (a) contain a single word or (b) comprise a valid subtree of a word and all its descendants in the dependency parse produced by the MST parser. This covers approximately 80% of arguments in the development data for both datasets.

The empty span, denoted $\emptyset$, is also included in $\mathcal{S}$, since some roles are not explicitly filled; in the SemEval 2007 development data, the average number of roles an evoked frame defines is 6.7, but the average number of overt arguments is only 1.7.[18] In training, if a labeled argument is not a subtree of the dependency parse, we add its span to $\mathcal{S}$.

[18]In the annotated data, each core role is filled with one of three types of *null instantiations* indicating how the role is conveyed implicitly. For instance, the imperative construction implicitly designates a role as filled by the addressee, and the corresponding filler is thus CNI (constructional null instantiation). In this work we do not distinguish different types of null instantiation. The interested reader may refer to Chen et al. (2010), who handle the different types of null instantions during argument identification.

Let $\mathcal{A}_i$ denote the mapping of roles in $\mathcal{R}_{f_i}$ to spans in $\mathcal{S}$. Our model makes a prediction for each $\mathcal{A}_i(r_k)$ (for all roles $r_k \in \mathcal{R}_{f_i}$) using:

$$\mathcal{A}_i(r_k) \leftarrow \operatorname*{argmax}_{s \in \mathcal{S}} p_{\boldsymbol{\psi}}(s \mid r_k, f_i, t_i, \mathbf{x}) \tag{5.4}$$

We use a conditional log-linear model over spans for each role of each evoked frame:

$$p_{\boldsymbol{\psi}}(\mathcal{A}_i(r_k) = s \mid f_i, t_i, \mathbf{x}) \quad = \quad \frac{\exp \boldsymbol{\psi}^\top \mathbf{h}(s, r_k, f_i, t_i, \mathbf{x})}{\displaystyle\sum_{s' \in \mathcal{S}} \exp \boldsymbol{\psi}^\top \mathbf{h}(s', r_k, f_i, t_i, \mathbf{x})} \tag{5.5}$$

Note that our model chooses the span for each role separately from the other roles and ignores all frames except the frame the role belongs to. Our model departs from the traditional SRL literature by modeling the argument identification problem in a single stage, rather than first classifying token spans as arguments and then labeling them. A constraint implicit in our formulation restricts each role to have at most one overt argument, which is consistent with 96.5% of the role instances in the SemEval 2007 training data and 96.4% of the role instances in the FrameNet 1.5 full text annotations.

Out of the overt argument spans in the training data, 12% are duplicates, having been used by some previous frame in the sentence (supposing some arbitrary ordering of frames). Our role-filling model, unlike a sentence-global argument detection-and-classification approach,[19] permits this sort of argument sharing among frames. Word tokens belong to an average of 1.6 argument spans, including the quarter of words that do not belong to any argument. Appending these local inference decisions together gives us the best mapping $\hat{\mathcal{A}}_t$ for target t. Features for our log-linear model (Equation 5.5) depend on the preprocessed sentence $\mathbf{x}$; the target t; a role r of frame f; and a candidate argument span $s \in \mathcal{S}$.[20] For features using the head word of the target t or a candidate argument span s, we use the heuristic described in footnote 11 for selecting the head of non-subtree spans.

Tables 5.6-5.7 lists the feature templates used in our model. Every feature template has a version which does not take into account the role being filled (so as to incorporate overall biases). The ◐ symbol indicates that the feature template also has a variant which is conjoined with r, the name of the role being filled; and ● indicates that the feature template additionally has a variant which is conjoined with both r and f, the name of the frame.[21] The role name–only variants provide for smoothing over frames for common types of roles such as Time and Place; see Matsubayashi et al. (2009) for

[19] J&N'07, like us, identify arguments for each target.

[20] In this section we use t, f, and r without subscripts since the features only consider a single role of a single target's frame.

[21] I.e., the ● symbol subsumes ◐, which in turn subsumes ○.

Features with both null and non-null variants: These features come in two flavors: if the argument is null, then one version fires; if it is overt (non-null), then another version fires.	
● some word in t has lemma λ	● some word in t has POS π
◐ some word in t has lemma λ, and the sentence uses PASSIVE voice	◐ some word in t has lemma λ, and the sentence uses ACTIVE voice
◐ the head of t has subcategorization sequence $\boldsymbol{\tau} = \langle \tau_1, \tau_2, \ldots \rangle$	◐ some syntactic dependent of the head of t has dependency type τ
● the head of t has c syntactic dependents	● bias feature (always fires)
Span content features: apply to overt argument candidates.	
○ POS tag π occurs for some word in s	
○ the head word of s has POS π	● $\|s\|$, the number of words in the candidate argument
○ the first word of s has POS π, provided $\|s\| > 0$	○ the head word of s has syntactic dependency type τ
○ the last word of s has POS π, provided $\|s\| > 0$	● the syntactic dependency type τ_{s_1} of the first word with respect to its head
● the first word of s: w_{s_1}, and its POS tag π_{s_1}, if π_{s_1} is a closed-class POS	● τ_{s_2}, provided that $\|s\| \geq 2$
● w_{s_2} and its closed-class POS tag π_{s_2}, provided that $\|s\| \geq 2$	● $\tau_{s_{\|s\|}}$, provided that $\|s\| \geq 3$
	○ the first word of s has lemma λ, provided $\|s\| > 0$
● the last word of s: $w_{s_{\|s\|}}$, and its closed-class POS tag $\pi_{s_{\|s\|}}$, provided that $\|s\| \geq 3$	○ the head word of s has lemma λ
	○ the last word of s has lemma λ, provided $\|s\| > 0$
◐ lemma λ is realized in some word in s	◐ lemma λ is realized in some word in s, the voice denoted in the span, s's position with respect to t (BEFORE, AFTER, or OVERLAPPING)
◐ lemma λ is realized in some word in s, the voice denoted in the span (ACTIVE or PASSIVE)	

Table 5.6: A basic set of argument identification features used in null and non-null spans and features looking at the content of a potential span. Section 5.4.1 describes the meanings of the different circles attached to each feature.

Syntactic features: apply to overt argument candidates.	
○ dependency path: sequence of labeled, directed edges from the head word of s to the head word of t	○ length of the dependency path
Span context POS features: for overt candidates, up to 6 of these features will be active.	
○ a word with POS π occurs up to 3 words before the first word of s	○ a word with POS π occurs up to 3 words after the last word of s
Ordering features: apply to overt argument candidates.	
● the position of s with respect to to the span of t: BEFORE, AFTER, or OVERLAPPING (i.e. there is at least one word shared by s and t)	○ target-argument crossing: there is at least one word shared by s and t, at least one word in s that is not in t, and at least one word in t that is not in s
○ linear word distance between the nearest word of s and the nearest word of t, provided s and t do not overlap	○ linear word distance between the middle word of s and the middle word of t, provided s and t do not overlap

Table 5.7: A second set of features used for argument identification. Section 5.4.1 describes the meanings of the different circles attached to each feature.

a detailed analysis of the effects of using role features at varying levels of granularity. Certain features in our model rely on closed-class POS tags, which are defined to be all Penn Treebank tags except for CD and tags that start with V, N, J, or R. Finally, the features that encode a count or a number are binned into groups: $(-\infty, -20]$, $[-19, -10]$, $[-9, -5]$, -4, -3, -2, -1, 0, 1, 2, 3, 4, $[5, 9]$, $[10, 19]$, $[20, \infty)$.

5.4.2 Training

We train the argument identification model by:

$$\max_{\boldsymbol{\psi}} \sum_{j=1}^{N} \sum_{i=1}^{m_j} \sum_{k=1}^{|\mathcal{R}_{f_i^{(j)}}|} \log p_{\boldsymbol{\psi}}(\mathcal{A}_i^{(j)}(r_k) \mid f_i^{(j)}, t_i^{(j)}, \mathbf{x}^{(j)}) - C\,\|\boldsymbol{\psi}\|_2^2 \tag{5.6}$$

The above objective function is concave. For experiments with the SemEval'07 data, we trained the model using stochastic gradient ascent (Bottou, 2003) with no Gaussian

Algorithm 1 Joint decoding of frame f_i's arguments. $\mathsf{top}_\mathsf{k}(\mathcal{S}, p_\psi, r_j)$ extracts the k most probable spans from $\mathcal{S}$, under p_ψ, for role r_j. $\mathsf{extend}(D^{0:(j-1)}, \mathcal{S}')$ extends each span vector in $D^{0:(j-1)}$ with the most probable non-overlapping span from $\mathcal{S}'$, resulting in k best extensions overall.

```
Input: $\mathsf{k} > 0$, $\mathcal{R}_{f_i}$, $\mathcal{S}$, the distribution $p_\psi$ from Equation 5.5 for each role $r_j \in \mathcal{R}_{f_i}$
Output: $\hat{\mathcal{A}}_i$, a high-scoring mapping of roles of $f_i$ to spans with no token overlap among
    the spans
 1: Calculate $\mathcal{A}_i$ according to Equation 5.4
 2: $\forall r \in \mathcal{R}_{f_i}$ such that $\mathcal{A}_i(r) = \emptyset$, let $\hat{\mathcal{A}}_i(r) \leftarrow \emptyset$
 3: $\mathcal{R}^+_{f_i} \leftarrow \{r : r \in \mathcal{R}_{f_i}, \mathcal{A}_i(r) \neq \emptyset\}$
 4: $\mathsf{n} \leftarrow |\mathcal{R}^+_{f_i}|$
 5: Arbitrarily order $\mathcal{R}^+_{f_i}$ as $\{r_1, r_2, \ldots r_\mathsf{n}\}$
 6: Let $D^{0:j} = \langle D^{0:j}_1, \ldots, D^{0:j}_\mathsf{k} \rangle$ refer to the k-best list of vectors of compatible filler spans
    for roles $r_1$ through $r_j$
 7: Initialize $D^{0:0}$ to be empty
 8: for $j = 1$ to n do
 9:     $D^{0:j} \leftarrow \mathsf{extend}(D^{0:(j-1)}, \mathsf{top}_\mathsf{k}(\mathcal{S}, p_\psi, r_j))$
10: end for
11: $\forall j \in \{1, \ldots, \mathsf{n}\}, \hat{\mathcal{A}}_i(r_j) \leftarrow D^{0:\mathsf{n}}_1[j]$
12: return $\hat{\mathcal{A}}_i$
```

regularization ($C = 0$).[22] Early stopping was done by tuning on the development set, and the best results were obtained with a batch size of 2 and 23 passes through the data.

On the FrameNet 1.5 release, we trained this model using L-BFGS (Liu and Nocedal, 1989) and ran it for 1000 iterations. C was tuned on the development data, and we obtained best results for $C = 1.0$. We did not use stochastic gradient descent for this dataset as the number of training samples increased and parallelization of L-BFGS on a multicore setup implementing MPI (Gropp et al., 1994) gave us faster training speeds.

5.4.3 Decoding with Beam Search

Naïve prediction of roles using Equation 5.4 may result in overlap among arguments filling different roles of a frame, since the argument identification model fills each role independently of the others. We want to enforce the constraint that two roles of a sin-

[22]This was the setting used by Das et al. (2010) and we kept it unchanged.

gle frame cannot be filled by overlapping spans.[23] Toutanova et al. (2005) presented a dynamic programming algorithm to prevent overlapping arguments for semantic role labeling; however, their approach used an orthogonal view to the argument identification stage, wherein they labeled phrase-structure tree constituents with semantic roles. This view helped them to adopt a dynamic programming approach, which does not suit our model because we find the best possible argument span for each role.

To eliminate illegal overlap, we adopt the beam search technique detailed in Algorithm 1. The algorithm produces a set of k-best hypotheses for a frame instance's full set of role-span pairs, but uses an approximation in order to avoid scoring an exponential number of hypotheses. After determining which roles are most likely not explicitly filled, it considers each of the other roles in turn: in each iteration, hypotheses incorporating a subset of roles are extended with high-scoring spans for the next role, always maintaining k alternatives. We set $k = 10000$ as beam width.[24]

5.4.4 Results

Performance of the argument identification model is presented in Table 5.8 for both datasets in consideration. We analyze them below.

SemEval'07 Data: For the dataset released for the SemEval shared task, the table shows how performance varies given different types of perfect input: both correct targets and correct frames, correct targets but automatically identified frames, and ultimately, no oracle input (the full frame parsing scenario). The first two rows of results isolate the argument identification task from the frame identification task. Given gold targets and frames, our argument identification model (without beam search) gets an F_1 score of 68.09%; when beam search is applied, this increases to 68.46%, with a noticeable increase in precision. Note that an estimated 19% of correct arguments are excluded because they are neither single words nor complete subtrees (see §5.4.1) of the automatic dependency parses.[25] Qualitatively, the problem of candidate span recall seems to be largely due to syntactic parse errors.[26] Although our upper bound performance is limited by errors when using the syntactic parse to determine candidate spans, our performance could

[23] On rare occasions a frame annotation may include a *secondary frame element layer*, allowing arguments to be shared among multiple roles in the frame; see Ruppenhofer et al. (2006) for details. The evaluation for this task only considers the primary layer, which is guaranteed to have disjoint arguments.

[24] We show the effect of varying beam widths in Table 5.9.

[25] Using all constituents from the 10-best syntactic parses would improve oracle recall of spans in the development set by just a couple of percentage points, at the computational cost of a larger pool of candidate arguments per role.

[26] Note that, because of our labels-only evaluation scheme (§5.1.3), arguments missing a word or containing an extra word receive no credit. In fact, of the frame roles correctly predicted as having an overt span, the correct span was predicted 66% of the time, while 10% of the time the predicted starting and ending boundaries of the span were off by a total of 1 or 2 words.

still improve; this suggests that the model has trouble discriminating between good and bad arguments, and that additional feature engineering or jointly decoding arguments of a sentence's frames may be beneficial.

The third and fourth rows show the effect of automatic frame identification on overall frame parsing performance. There is a 22% decrease in F_1 (18% when partial credit is given for related frames), suggesting that improved frame identification or joint prediction of frames and arguments is likely to have a sizeable impact on overall performance. The final two rows of the SemEval 2007 section of the table compare our full model (target, frame, and argument identification) with the baseline, showing significant improvement of more than 4.4 F_1 points for both exact and partial frame matching. As with frame identification, we compared the argument identification stage with that of J&N'07 in isolation, using the automatically identified targets and frames from the latter as input to our model. As shown in the 5th row of the table, with partial frame matching, this gave us an F_1 score of 48.1% on the test set—significantly better ($p < 0.05$) than 45.6%, the full parsing result from J&N'07 (6th row in Table 5.8). This indicates that our argument identification model—which uses a single discriminative model with a large number of features for role filling (rather than argument labeling)—is more accurate than the previous state of the art.

FrameNet 1.5 Release: The last three rows show results on the newer dataset, which is part of the FrameNet 1.5 release. Like in the frame identification results of Table 5.5, we do not show results using predicted targets, as we only test the performance of the statistical models. First, we observe, that for results with gold frames, the F_1 score is 79.08% with naïve decoding, which is significantly higher in comparison with the SemEval counterpart. This indicates that with increased data, performance on the task gets much better. We also observe that beam search improves precision by nearly 2%, while getting rid of overlapping arguments. Finally, when both model frames and model arguments are used, we get an F_1 score of 68.45%, which is encouraging in comparison to the best results we achieved on the SemEval 2007 dataset.

5.5 Collective Argument Identification

The argument identification strategy described in the previous section does not capture some facets of semantic knowledge represented declaratively in FrameNet. Here, we present an approach that exploits such knowledge in a principled, unified, and intuitive way. In prior research using FrameNet, these interactions have been largely ignored, though they have the potential to improve the quality and consistency of semantic analysis. The beam search technique (Algorithm 1) handles constraints in the form of avoiding argument overlaps, but is greedy and cannot handle other forms of constraints.

ARGUMENT IDENTIFICATION					exact frame matching			partial frame matching		
		targets	*frames*	*decoding*	*P*	*R*	F_1	*P*	*R*	F_1
SemEval'07 Data	Argument identification (full)	gold	gold	naïve	77.43	60.76	68.09			
		gold	gold	beam	78.71	60.57	68.46			
	Parsing (oracle targets)	gold	model	beam	49.68	42.82	46.00	57.85	49.86	53.56
	Parsing (full)	auto	model	beam	**58.08**	**38.76**	**46.49**	**62.76**	**41.89**	**50.24**
	Parsing (J&N'07 targets and frames)	auto	model	beam	56.26	36.63	44.37	60.98	39.70	48.09
	Baseline: J&N'07	*auto*	*model*	*N/A*	*51.59*	*35.44*	*42.01*	*56.01*	*38.48*	*45.62*
FrameNet 1.5 Release	Argument identification (full)	gold	gold	naïve	82.00	76.36	79.08			
		gold	gold	beam	83.83	76.28	79.88			
	Parsing (oracle targets)	gold	model	beam	67.81	60.68	64.05	72.47	64.85	68.45

Table 5.8: Argument identification results on both the SemEval'07 data as well as the full text annotations of FrameNet 1.5. For decoding, "beam" and "naïve" indicate whether the approximate joint decoding algorithm has been used or local independent decisions have been made for argument identification, respectively. For full parsing, bolded scores indicate significant improvements relative to the baseline ($p < 0.05$).

Here, we present an algorithm that identifies the full collection of arguments of a target given its semantic frame. Although we work within the conventions of FrameNet, our approach is generalizable to other semantic role labeling (SRL) frameworks. We model argument identification as constrained optimization, where the constraints come from expert knowledge encoded in FrameNet. Following prior work on PropBank-style SRL (Kingsbury and Palmer, 2002) that dealt with similar constrained problems (Punyakanok et al., 2004; Roth and Yih, 2004, *inter alia*), we incorporate this declarative knowledge in an integer linear program (ILP).

Because general-purpose ILP solvers are proprietary and do not fully exploit the structure of the problem, we turn to a class of optimization techniques called **dual decomposition** (Komodakis et al., 2007; Rush et al., 2010; Martins et al., 2011a). We derive a modular, extensible, parallelizable approach in which semantic constraints map not just to declarative components in the algorithm, but also to procedural ones, in the form of "workers." While dual decomposition algorithms only solve a relaxation of the original problem, we make our approach *exact* by wrapping the algorithm in a branch-and-bound search procedure.

We experimentally find that our algorithm achieves accuracy comparable to a state-of-the-art system, while respecting all imposed linguistic constraints. In comparison to beam search that violates many of these constraints, the presented exact decoder is slower, but it decodes nine times faster than CPLEX, a state-of-the-art, proprietary, general-purpose exact ILP solver.

5.5.1 Background

Most accurate SRL systems that use conventions from PropBank (Kingsbury and Palmer, 2002) and NomBank (Meyers et al., 2004) employ joint inference for semantic role labeling (Màrquez et al., 2008). To our knowledge, the separate line of work investigating frame-semantic parsing has not dealt with non-local information until this work. A common trait in prior work, both in PropBank and FrameNet conventions, has been the use of a two-stage model that identifies arguments first, then labels them, often using joint inference techniques like dynamic programming or integer linear programs. As mentioned before, we treat both problems together here.

Solving inference in NLP problems using LP relaxations is becoming increasingly popular. While early work has focused on declarative formulations tackled with off-the-shelf solvers (Martins et al., 2009, 2010), Rush et al. (2010) proposed subgradient-based dual decomposition (also called Lagrangian relaxation) as a way of exploiting the structure of the problem and reusing existing combinatorial algorithms. The method allows the combination of models which are tractable individually, but not jointly, by solving a relaxation of the original problem. Since then, dual decomposition has been used to build more accurate models for dependency parsing (Koo et al., 2010), CCG supertagging and parsing (Auli and Lopez, 2011) and machine translation (DeNero and Macherey, 2011; Rush and Collins, 2011; Chang and Collins, 2011).

Recently, Martins et al. (2011b) showed that the success of subgradient-based dual decomposition strongly relies on breaking down the original problem into a "good" decomposition, *i.e.*, one with few overlapping components. This leaves out many declarative constrained problems, for which such a good decomposition is not readily available. For those, Martins et al. (2011b) proposed the **AD**^{3} algorithm, which retains the modularity of previous methods, but can handle thousands of small overlapping components. We adopt that algorithm as it perfectly suits the problem of argument identification, as we observe in the following sections.

We also contribute an exact branch-and-bound technique wrapped around AD^3. A related line of research is that of Rush and Collins (2011), who proposed a tightening procedure for dual decomposition, which can be seen as a cutting plane method (another popular approach in combinatorial optimization). That procedure would involve constructing larger factors, hence is not a good match for AD^3, which works best with smaller factors/constraints as found in our problem.

5.5.2 Joint Inference

Here, we take a declarative approach to modeling argument identification using an ILP and relate our formulation to prior work in shallow semantic parsing. We show how knowledge specified in a linguistic resource (which is FrameNet in our case) can be used to derive the constraints used in our ILP. Finally, we draw connections of our specification to graphical models, a popular formalism in AI, and describe how the constraints can be treated as factors in a factor graph.

Declarative Specification

Let us simplify notation by considering a given target t and not consider its index in a sentence $\mathbf{x}$; let the semantic frame it evokes be f. To solely evaluate argument identification, we assume that the semantic frame f is given, which is traditionally the case in controlled experiments used to evaluate SRL systems (Màrquez et al., 2008). Let the set of roles associated with the frame f be $\mathcal{R}_f$. In sentence $\mathbf{x}$, the set of candidate spans of words that might fill each role is enumerated, usually following an overgenerating heuristic, which is described in §5.4.1; as before, we call this set of spans $\mathcal{S}$. This set also includes the null span $\emptyset$; connecting it to a role $r \in \mathcal{R}_f$ denotes that the role is not overt. Our approach assumes a scoring function that gives a strength of association between roles and candidate spans. For each role $r \in \mathcal{R}_f$ and span $s \in \mathcal{S}$, this score is parameterized as:

$$c(r, s) = \boldsymbol{\psi}^\top \mathbf{h}(s, r, f, t, \mathbf{x}), \tag{5.7}$$

where ψ are model weights and $\mathbf{h}$ is a feature function that looks at the target t, the evoked frame f, sentence $\mathbf{x}$, and its syntactic analysis, along with r and s. This scoring function is identical in form to the numerator's exponent in the log-linear model described in Equation 5.5. The SRL literature provides many feature functions of this form and many ways to use machine learning to acquire ψ. Our presented method does not make any assumptions about the score except that it has the form in Equation 5.7.

We define a vector $\mathbf{z}$ of binary variables $z_{r,s} \in \{0,1\}$ for every role and span pair. We have that: $\mathbf{z} \in \{0,1\}^d$, where $d = |\mathcal{R}_f| \times |\mathcal{S}|$. $z_{r,s} = 1$ means that role r is filled by span s. Given the binary $\mathbf{z}$ vector, it is straightforward to recover the collection of arguments by checking which components $z_{r,s}$ have an assignment of 1; we use this strategy to find arguments, as described in §5.5.4 (strategies 4 and 6). The joint argument identification task can be represented as a constrained optimization problem:

$$\begin{aligned} \text{maximize} \quad & \sum_{r \in \mathcal{R}_f} \sum_{s \in \mathcal{S}} c(r,s) \times z_{r,s} \\ \text{with respect to} \quad & \mathbf{z} \in \{0,1\}^d \\ \text{such that} \quad & \mathbf{A}\mathbf{z} \le \mathbf{b}. \end{aligned} \tag{5.8}$$

The last line imposes constraints on the mapping between roles and spans; these are motivated on linguistic grounds and are described next.[27]

1. **Uniqueness:** Each role r is filled by at most one span in $\mathcal{S}$. This constraint can be expressed by:
$$\forall r \in \mathcal{R}_f, \sum_{s \in \mathcal{S}} z_{r,s} = 1. \tag{5.9}$$
There are $O(|\mathcal{R}_f|)$ such constraints. Note that since $\mathcal{S}$ contains the null span $\emptyset$, non-overt roles are also captured using the above constraints. Such a constraint is used extensively in prior literature (Punyakanok et al., 2004, §4.1).

2. **Overlap:** SRL systems commonly constrain roles to be filled by non-overlapping spans. For example, Toutanova et al. (2005) used dynamic programming over a phrase structure tree to prevent overlaps between arguments, and Punyakanok et al. (2004) used constraints in an ILP to respect this requirement. Inspired by the latter, we require that each input sentence position of $\mathbf{x}$ be covered by at most one argument. For each role $r \in \mathcal{R}_f$, we define:
$$\mathcal{G}_r(i) = \{s \mid s \in \mathcal{S}, s \text{ covers position } i \text{ in } \mathbf{x}\}. \tag{5.10}$$

[27] Note that equality constraints $\mathbf{a} \cdot \mathbf{z} = b$ can be transformed into double-side inequalities $\mathbf{a} \cdot \mathbf{z} \le b$ and $-\mathbf{a} \cdot \mathbf{z} \le -b$.

We can define our overlap constraints in terms of $\mathcal{G}_r$ as follows, for every sentence position i:

$$\forall i \in \{1, \dots, |\mathbf{x}|\}, \quad \sum_{r \in \mathcal{R}_f} \sum_{s \in \mathcal{G}_r(i)} z_{r,s} \leq 1, \tag{5.11}$$

This gives us $O(|\mathbf{x}|)$ constraints.

3. **Pairwise "Exclusions":** For many target classes, there are pairs of roles forbidden to appear together in the analysis of a single target token. Consider the following two sentences:

$$\underbrace{\text{A blackberry}}_{\text{Entity\_1}} \textbf{ resembles } \underbrace{\text{a loganberry}}_{\text{Entity\_2}}. \tag{5.12}$$

$$\underbrace{\text{Most berries}}_{\text{Entities}} \textbf{ resemble } \text{each other.} \tag{5.13}$$

Consider the uninflected target **resemble** in both sentences, evoking the same meaning. In example 5.12, two roles, which we call Entity_1 and Entity_2 describe two entities that are similar to each other. In the second sentence, a phrase fulfills a third role, called Entities, that collectively denotes some objects that are similar. It is clear that the roles Entity_1 and Entities cannot be overt for the same target at once, because the latter already captures the function of the former; a similar argument holds for the Entity_2 and Entities roles. We call this phenomenon the "excludes" relationship. Let us define a set of pairs from $\mathcal{R}_f$ that have this relationship:

$$\mathit{Excl}_f = \{(r_i, r_j) \mid r_i \text{ and } r_j \textit{ exclude} \text{ each other}\}$$

Using the above set, we define the constraint:

$$\forall (r_i, r_j) \in \mathit{Excl}_f, \; z_{r_i,\emptyset} + z_{r_j,\emptyset} \geq 1 \tag{5.14}$$

In English: if both roles are overt in a parse, this constraint will be violated, and we will not respect the "excludes" relationship between the pair. If neither or only one of the roles is overt, the constraint is satisfied. The total number of such constraints is $O(|\mathit{Excl}_f|)$, which is the number of pairwise "excludes" relationships of a given frame.

4. **Pairwise "Requirements":** The sentence in example 5.12 illustrates another kind of constraint. The target **resemble** cannot have only one of Entity_1 and Entity_2 as roles in text. For example,

$$\text{* } \underbrace{\text{A blackberry}}_{\text{Entity\_1}} \textbf{ resembles}. \tag{5.15}$$

Enforcing the overtness of two roles sharing this "requires" relationship is straightforward. We define the following set for a frame f:

$$\mathit{Req}_f = \{(r_i, r_j) \mid r_i \text{ and } r_j \textit{ require} \text{ each other}\}$$

This leads to constraints of the form

$$\forall (r_i, r_j) \in \mathit{Req}_f,\; z_{r_i,\emptyset} - z_{r_j,\emptyset} = 0 \tag{5.16}$$

If one role is overt (or absent), so must the other be. A related constraint has been used previously in the SRL literature, enforcing joint overtness relationships between core arguments and referential arguments (Punyakanok et al., 2004, §4.1), which are formally similar to the example above.[28]

Integer Linear Program and Relaxation

Plugging the constraints in Eqs. 5.9, 5.11, 5.14 and 5.16 into the last line of Equation 5.8, we have the argument identification problem expressed as an ILP, since the indicator variables $\mathbf{z}$ are binary. Here, apart from the ILP formulation, we will consider the following *relaxation* of Equation 5.8, which replaces the binary constraint $\mathbf{z} \in \{0,1\}^d$ by a unit interval constraint $\mathbf{z} \in [0,1]^d$, yielding a *linear* program:

$$\begin{aligned} \text{maximize} \quad & \sum_{r \in \mathcal{R}_f} \sum_{s \in \mathcal{S}} c(r,s) \times z_{r,s} \\ \text{with respect to} \quad & \mathbf{z} \in [0,1]^d \\ \text{such that} \quad & \mathbf{A}\mathbf{z} \leq \mathbf{b}. \end{aligned} \tag{5.17}$$

There are several LP and ILP solvers available, and a great deal of effort has been spent by the optimization community to devise efficient generic solvers. An example is CPLEX, a state-of-the-art solver for mixed integer programming that we employ as a baseline to solve the ILP in Equation 5.8 as well as its LP relaxation in Equation 5.17. Like many of the best implementations, CPLEX is proprietary.

[28] We noticed in the annotated data, in some cases, the "requires" constraint is violated by the FrameNet annotators. This happens mostly when one of the required roles is absent in the sentence containing the target, but is rather instantiated in an earlier sentence (Gerber and Chai, 2010). We apply the hard constraint in Equation 5.16, though extending our algorithm to seek arguments outside the sentence is straightforward.

Linguistic Constraints from FrameNet

Although enforcing the four different sets of constraints above is intuitive from a general linguistic perspective, we ground their use in definitive linguistic information present in the FrameNet lexicon. From the annotated data in the FrameNet 1.5 release, we gathered that only 3.6% of the time is a role instantiated multiple times by different spans in a sentence. This justifies the uniqueness constraint enforced by Equation 5.9. Use of such a constraint is also consistent with prior work in frame-semantic parsing (Johansson and Nugues, 2007). Similarly, we found that in the annotations, no arguments overlapped with each other for a given target. Hence, the overlap constraints in Equation 5.11 are also justified.

Our third and fourth sets of constraints, presented in Eqs. 5.14 and 5.16, come from FrameNet, too. Examples 5.12–5.13 are instances where the target **resemble** evokes the SIMILARITY frame, which is defined in FrameNet as:

> Two or more distinct entities, which may be concrete or abstract objects or types, are characterized as being similar to each other. Depending on figure/ground relations, the entities may be expressed in two distinct frame elements and constituents, Entity_1 and Entity_2, or jointly as a single frame element and constituent, Entities.

For this frame, the lexicon lists several roles other than the three roles we have already observed, such as Dimension (the dimension along which the entities are similar), Differentiating_fact (a fact that reveals how the concerned entities are similar or different), and so forth. Along with the roles, FrameNet also declares the "excludes" and "requires" relationships noted in our discussion in Section 5.5.2. The case of the SIMILARITY frame is not unique; in Fig. 5.1, the frame COLLABORATION, evoked by the target **partners**, also has two roles Partner_1 and Partner_2 that share the "requires" relationship. In fact, out of 877 frames in FrameNet 1.5, 204 frames have at least a pair of roles for which the "excludes" relationship holds, and 54 list at least a pair of roles that share the "requires" relationship.

Constraints as Factors in a Graphical Model

The LP in Equation 5.17 can be represented as a maximum *a posteriori* (MAP) inference problem in an undirected graphical model. In the factor graph, each component of $\mathbf{z}$ corresponds to a binary variable, and each instantiation of a constraint in Equations 5.9, 5.11, 5.14 and 5.16 corresponds to a factor. Smith and Eisner (2008) and Martins et al. (2010) used such a representation to impose constraints in a dependency parsing problem; the latter discussed the equivalence of linear programs and factor graphs for representing discrete optimization problems. Each of our constraints take standard factor forms we can describe using the terminology of Smith and Eisner (2008) and Martins et al. (2010). The uniqueness constraint in Equation 5.9 corresponds to an XOR factor, while the overlap constraint in Equation 5.11 corresponds to an ATMOSTONE factor. The constraints in Equation 5.14 enforcing the "excludes" relationship can be represented with an OR factor. Finally, each "requires" constraints in Equation 5.16 is equivalent to an XORWITHOUTPUT factor.

In the following section, we describe how we arrive at solutions for the LP in Equation 5.17 using dual decomposition, and how we adapt it to efficiently recover the *exact* solution of the ILP (Equation 5.8), without the need of an off-the-shelf ILP solver.

5.5.3 "Augmented" Dual Decomposition

Dual decomposition methods address complex optimization problems in the dual, by dividing them into simple worker problems, which are repeatedly solved until a consensus is reached. The most simple technique relies on the subgradient algorithm (Komodakis et al., 2007; Rush et al., 2010); as an alternative, an augmented Lagrangian technique was proposed by Martins et al. (2011a,b), which is more suitable when there are many small components—commonly the case in declarative constrained problems, such as the one at hand. Here, we present a brief overview of the latter, which is called *Alternating Direction Dual Decomposition* (AD^3).

Let us start by establishing some notation. Let $m \in \{1, \ldots, M\}$ index a factor, and denote by $\mathbf{i}(m)$ the vector of indices of variables linked to that factor. (Recall that each factor represents the instantiation of a constraint.) We introduce a new set of variables, $\mathbf{u} \in \mathbb{R}^d$, called the "witness" vector. We split the vector $\mathbf{z}$ into M overlapping pieces $\mathbf{z}_1, \ldots, \mathbf{z}_M$, where each $\mathbf{z}_m \in [0,1]^{|\mathbf{i}(m)|}$, and add M constraints $\mathbf{z}_m = \mathbf{u}_{\mathbf{i}(m)}$ to impose that all the pieces must agree with the witness (and therefore with each other). Each of the M constraints described in §5.5.2 can be encoded with its own matrix $\mathbf{A}_m$ and vector $\mathbf{b}_m$ (which jointly define $\mathbf{A}$ and $\mathbf{b}$ in Equation 5.17). For convenience, we denote by $\mathbf{c} \in \mathbb{R}^d$ the score vector, whose components are $c(r,s)$, for each $r \in \mathcal{R}_f$ and $s \in \mathcal{S}$ (Equation 5.7), and define the following scores for the mth subproblem:

$$c_m(r,s) = \delta(r,s)^{-1} c(r,s), \ \forall (r,s) \in \mathbf{i}(m), \tag{5.18}$$

where $\delta(r,s)$ is the number of constraints that involve role r and span s. Note that according to this definition, $\mathbf{c} \cdot \mathbf{z} = \sum_{m=1}^{M} \mathbf{c}_m \cdot \mathbf{z}_m$. We can rewrite the LP in Equation 5.17

in the following equivalent form:

$$\begin{array}{rl} \text{maximize} & \displaystyle\sum_{m=1}^{M} \mathbf{c}_m \cdot \mathbf{z}_m \\ \text{with respect to} & \mathbf{u} \in \mathbb{R}^d,\ \mathbf{z}_m \in [0,1]^{\mathbf{i}(m)}, \quad \forall m \\ \text{such that} & \mathbf{A}_m \mathbf{z}_m \leq \mathbf{b}_m, \quad \forall m \\ & \mathbf{z}_m = \mathbf{u}_{\mathbf{i}(m)}, \quad \forall m. \end{array} \tag{5.19}$$

We next augment the objective with a quadratic penalty term $\frac{\rho}{2}\sum_{m=1}^{M}\|\mathbf{z}_m - \mathbf{u}_{\mathbf{i}(m)}\|^2$ (for some $\rho > 0$). This does not affect the solution of the problem, since the equality constraints in the last line force this penalty to vanish. However, as we will see, this penalty will influence the workers and will lead to faster consensus. Next, we introduce Lagrange multipliers $\boldsymbol{\lambda}_m$ for those equality constraints, so that the augmented Lagrangian function becomes:

$$\begin{aligned} L_\rho(\mathbf{z}, \mathbf{u}, \boldsymbol{\lambda}) \;=\; & \sum_{m=1}^{M} (\mathbf{c}_m + \boldsymbol{\lambda}_m) \cdot \mathbf{z}_m - \boldsymbol{\lambda}_m \cdot \mathbf{u}_{\mathbf{i}(m)} \\ & - \frac{\rho}{2}\|\mathbf{z}_m - \mathbf{u}_{\mathbf{i}(m)}\|^2. \end{aligned} \tag{5.20}$$

The AD^3 algorithm seeks a saddle point of L_ρ by performing alternating maximization with respect to $\mathbf{z}$ and $\mathbf{u}$, followed by a gradient update of $\boldsymbol{\lambda}$. The result is shown as Algorithm 2. Like dual decomposition approaches, it repeatedly performs a *broadcast* operation (the $\mathbf{z}_m$-updates, which can be done in parallel, one constraint per "worker") and a *gather* operation (the $\mathbf{u}$- and $\boldsymbol{\lambda}$-updates). Each $\mathbf{u}$-operation can be seen as an averaged voting which takes into consideration each worker's results.

Like in the subgradient method, the $\boldsymbol{\lambda}$-updates can be regarded as price adjustments, which will affect the next round of $\mathbf{z}_m$-updates. The only difference with respect to the subgradient method (Rush et al., 2010) is that each subproblem involved in a $\mathbf{z}_m$-update also has a quadratic penalty that penalizes deviations from the previous average voting; it is this term that accelerates consensus and therefore convergence. Martins et al. (2011b) also provide stopping criteria for the iterative updates using primal and dual residuals that measure convergence; we refer the reader to that paper for details.

A key attraction of this algorithm is all the components of the declarative specification remain intact in the procedural form. Each worker corresponds exactly to one constraint in the ILP, which corresponds to one linguistic constraint. There is no need to work out *when*, during the procedure, each constraint might have an effect, as in beam search.

Algorithm 2 AD^3 for Argument Identification

```
1: input:
     • role-span matching scores $\mathbf{c} := \langle c(r,s) \rangle_{r,s}$,
     • structural constraints $\langle \mathbf{A}_m, \mathbf{b}_m \rangle_{m=1}^{M}$,
     • penalty $\rho > 0$
2: initialize $t \leftarrow 1$
3: initialize $\mathbf{u}^1$ uniformly (i.e., $u(r,s) = 0.5, \ \forall r, s$)
4: initialize each $\boldsymbol{\lambda}_m^1 = \mathbf{0}, \ \forall m \in \{1, \ldots, M\}$
5: repeat
6:     for each $m = 1, \ldots, M$ do
7:         make a $\mathbf{z}_m$-update by finding the best scoring analysis for the $m$th constraint,
           with penalties for deviating from the consensus $\mathbf{u}$:
           $$\mathbf{z}_m^{(t+1)} \leftarrow \operatorname*{argmax}_{\mathbf{A}_m \mathbf{z}_m^t \leq \mathbf{b}_m} (\mathbf{c}_m + \boldsymbol{\lambda}_m^t) \cdot \mathbf{z}_m - \frac{\rho}{2} \|\mathbf{z}_m - \mathbf{u}_{\mathbf{i}(m)}^t\|^2 \qquad (5.21)$$
8:     end for
9:     make a $\mathbf{u}$-update by updating the consensus solution, averaging $\mathbf{z}_1, \ldots, \mathbf{z}_m$:
       $$u^{(t+1)}(r,s) \leftarrow \frac{1}{\delta(r,s)} \sum_{m:(r,s) \in \mathbf{i}(m)} z_m^{(t+1)}(r,s)$$
10:    make a $\boldsymbol{\lambda}$-update:
       $$\boldsymbol{\lambda}_m^{(t+1)} \leftarrow \boldsymbol{\lambda}_m^t - \rho(\mathbf{z}_m^{(t+1)} - \mathbf{u}_{\mathbf{i}(m)}^{(t+1)}), \quad \forall m$$
11:    $t \leftarrow t + 1$
12: until convergence.
13: output: relaxed primal solution $\mathbf{u}^*$ and dual solution $\boldsymbol{\lambda}^*$. If $\mathbf{u}^*$ is integer, it will
    encode an assignment of spans to roles. Otherwise, it will provide an upper bound
    of the true optimum.
```

Solving the subproblems

Here, we consider the procedures used to solve the subproblems, corresponding to each $\mathbf{z}_m$-update, as shown in Equation 5.21. These subproblems can be solved efficiently for

Algorithm 3 Projection onto the simplex

```
1: input: $\langle a_1, a_2, \ldots, a_n \rangle$
2: Sort $\langle a_1, a_2, \ldots, a_n \rangle$ into $\langle b_1, b_2, \ldots, b_n \rangle$: $b_1 \geq b_2 \cdots \geq b_n$
3: Find $\alpha = \max \left\{ j \in [n] \mid b_j - \frac{1}{j} \big( \sum_{k=1}^{j} b_k - 1 \big) > 0 \right\}$
4: Define $\tau = \frac{1}{\alpha} \left( \sum_{k=1}^{\alpha} b_k - 1 \right)$
5: output: $\langle z_1, z_2, \ldots, z_n \rangle$ with $z_i = \max\{a_i - \tau, 0\}$.
```

several cases that arises in language processing tasks, and here we consider the four specific $\mathbf{z}_m$-subproblems associated with the XOR, XORWITHOUTPUT, OR and ATMOSTONE factors. Following Martins et al. (2011b), we transform Equation 5.21 to the following generic procedure for each of the above types of subproblems:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} \|\mathbf{z}_m - \mathbf{a}_{\mathbf{i}(m)}\|_2^2 \\ \text{with respect to} \quad & \mathbf{z}_m \in [0,1]^{\mathbf{i}(m)} \\ \text{such that} \quad & \mathbf{A}_m \mathbf{z}_m \leq \mathbf{b}_m. \end{aligned} \tag{5.22}$$

In the above equation, we ignore timestep t in the superscript to simplify notation. $\mathbf{a}_{\mathbf{i}(m)}$ in the above equation is defined as:

$$\mathbf{a}_{\mathbf{i}(m)} = \mathbf{u}_{\mathbf{i}(m)} + \rho^{-1}(\mathbf{c}_m + \boldsymbol{\lambda}_m) \tag{5.23}$$

We consider the four different factors relevant to our problem at a time as follows.

1. XOR **Factor** (the "uniqueness" constraints): The first factor, which employs the uniqueness or the XOR constraint, can be specified as:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} \|\mathbf{z}_m - \mathbf{a}_{\mathbf{i}(m)}\|_2^2 \\ \text{with respect to} \quad & \mathbf{z}_m \in [0,1]^{\mathbf{i}(m)} \\ \text{such that} \quad & \|\mathbf{z}_m\|_1 = 1. \end{aligned} \tag{5.24}$$

   The above problem can be solved by computing a projection onto the probability simplex as denoted by Algorithm 3 (Duchi et al., 2008). Given the dimensionality $|\mathbf{i}(m)|$ of $\mathbf{z}_m$, this projection operation is just a sort which takes $O(|\mathbf{i}(m)| \log |\mathbf{i}(m)|)$ time.

2. XORWITHOUTPUT **Factor** (the "requires" constraints): This factor can be expressed as the procedure:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\mathbf{z}_m - \mathbf{a}_{\mathbf{i}(m)}\|_2^2 \\ \text{with respect to} \quad & \mathbf{z}_m \in [0,1]^{\mathbf{i}(m)} \\ \text{such that} \quad & \sum_{k=1,k\neq j}^{|\mathbf{i}(m)|} z_{m,k} = z_{m,j}. \end{aligned} \tag{5.25}$$

   The above can be solved by using the procedure above by using the XOR procedure itself but using the following twist:

   (a) Set $a'_{m,j} = 1 - a_{m,j}$, where $a_{m,j}$ is the component of $\mathbf{a}_{\mathbf{i}(m)}$ that corresponds to $z_{m,j}$ in Equation 5.25. Set $a'_{m,k} = a_{m,k}$, $\forall k, k \neq j$.
   (b) Obtain $\langle z'_{m,1}, z'_{m,2}, \ldots, z'_{m,|\mathbf{i}(m)|}\rangle$ by using Algorithm 3 with $\langle a'_{m,1}, a'_{m,2}, \ldots, a'_{m,|\mathbf{i}(m)|}\rangle$ input.
   (c) Set $z_{m,j} = 1 - z'_{m,j}$ and $\forall k \neq j, z_{m,j} = z'_{m,j}$.

3. OR **Factor** (the "excludes" constraints): The OR factor can be expressed as:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\mathbf{z}_m - \mathbf{a}_{\mathbf{i}(m)}\|_2^2 \\ \text{with respect to} \quad & \mathbf{z}_m \in [0,1]^{\mathbf{i}(m)} \\ \text{such that} \quad & \|\mathbf{z}_m\|_1 \geq 1. \end{aligned} \tag{5.26}$$

   This factor can be computed as:

   (a) Set $\forall k \in \mathbf{i}(m), z_{m,k} = \min\{\max\{a_{m,k}, 0\}, 1\}$.
   (b) If $\|\mathbf{z}_m\|_1 \geq 1$, return $\langle z_{m,1}, \ldots, z_{m,|\mathbf{i}(m)|}\rangle$. Else, project $\langle a_{m,1}, \ldots, a_{m,|\mathbf{i}(m)|}\rangle$ to the simplex.

   The runtime of the above procedure is also $O(|\mathbf{i}(m)| \log |\mathbf{i}(m)|)$. A proof of correctness can be found in Appendix B of Martins et al. (2011b).

4. ATMOSTONE **Factor** (the "overlap" constraints): This factor can be represented as:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\mathbf{z}_m - \mathbf{a}_{\mathbf{i}(m)}\|_2^2 \\ \text{with respect to} \quad & \mathbf{z}_m \in [0,1]^{\mathbf{i}(m)} \\ \text{such that} \quad & \|\mathbf{z}_m\|_1 \leq 1. \end{aligned} \tag{5.27}$$

   It can be computed in a similar fashion as the OR factor:

(a) Set $\forall k \in \mathbf{i}(m), z_{m,k} = \min\{\max\{a_{m,k}, 0\}, 1\}$.

(b) If $\|\mathbf{z}_m\|_1 \leq 1$, return $\langle z_{m,1}, \ldots, z_{m,|\mathbf{i}(m)|}\rangle$. Else, project $\langle a_{m,1}, \ldots, a_{m,|\mathbf{i}(m)|}\rangle$ to the simplex.

The runtime of this procedure is also the same as the OR factor, and the proof of this procedure's correctness follows from the proof in Appendix B of Martins et al. (2011b).

Caching

As mentioned by Martins et al. (2011b), as the algorithm comes close to convergence, many subproblems become unchanged and their solutions can be cached. By caching the subproblems, we managed to reduce runtime by about 60%.

Exact decoding

It is worth recalling that AD^3, like other dual decomposition algorithms, solves a *relaxation* of the actual problem. Although we have observed that the relaxation is often tight—cf. §5.5.4—this is not always the case. Specifically, a fractional solution may be obtained, which is not interpretable as an argument, and therefore it is desirable to have a strategy to recover the exact solution. Two observations are noteworthy. First, the optimal value of the relaxed problem (Equation 5.17) provides an upper bound to the original problem (Equation 5.8). This is because Equation 5.8 has the additional integer constraint on the variables. In particular, any feasible dual point provides an upper bound to the original problem's optimal value. Second, during execution of the AD^3 algorithm, we always keep track of a sequence of feasible dual points. Therefore, each iteration constructs tighter and tighter upper bounds. With this machinery, we have all that is necessary for implementing a branch-and-bound search that finds the exact solution of the ILP. The procedure works recursively as follows:

1. Initialize $L = -\infty$ (our best value so far).

2. Run Algorithm 2. If the solution $\mathbf{u}^*$ is integer, return $\mathbf{u}^*$ and set L to the objective value. If along the execution we obtain an upper bound less than L, then Algorithm 2 can be safely stopped and return "infeasible"—this is the *bound* part. Otherwise (if $\mathbf{u}^*$ is fractional) go to step 3.

3. Find the "most fractional" component of $\mathbf{u}^*$ (call it u_j^*) and *branch*: constrain $u_j = 0$ and go to step 2, eventually obtaining an integer solution $\mathbf{u}_0^*$ or infeasibility; and then constrain $u_j = 1$ and do the same, obtaining $\mathbf{u}_1^*$. Return the $\mathbf{u}^* \in \{\mathbf{u}_0^*, \mathbf{u}_1^*\}$ that yields the largest objective value.

Although this procedure may have worst-case exponential runtime, we found it empirically to obtain the exact solution in all test cases.

5.5.4 Results with Collective Argument Identification

We present experiments only on argument identification in this section, as our goal is to exhibit the importance of incorporating the various linguistic constraints during our inference procedure. We present results on the full text annotations of FrameNet 1.5, and do not experiment on the SemEval 2007 benchmark, as we have already observed that constraint-agnostic models are superior than prior work. We trained the model weight $\boldsymbol{\psi}$ used in the scoring function c, in the same way as in §5.4.1, i.e. by training a logistic regression model using maximum conditional log-likelihood. The AD^3 parameter ρ was initialized to 0.1, we followed Martins et al. (2011b) in dynamically adjusting it to keep a balance between the primal and dual residuals.

We compare the following algorithms to demonstrate the efficacy of our collective argument identification approach:

1. **Local**: this is a naïve argument identification strategy that selects the best span for each role r, according to the score function $c(r, s)$. The idea is exactly similar to Equation 5.4 as described in §5.4.1. It ignores all constraints except "uniqueness." This also corresponds to the results seen in the 7th row of Table 5.8.

2. **Beam**: this strategy employs greedy beam search to eliminate overlaps between predicted arguments, as described in Algorithm 1. Note that it does not try to respect the "excludes" and "requires" constraints between pairs of roles. The default size of the beam in §1 was a safe 10,000; this resulted in extremely slow decoding times. For time comparison, we tried beam sizes of 100 and 2 (the latter being the smallest size that achieves the same F_1 score on the FrameNet 1.5 dev set.) This, with beam size 10,000 corresponds to the 8th row of Table 5.8.

3. **CPLEX**, *LP*: this uses CPLEX to solve the relaxed LP in Equation 5.17. To handle fractional $\mathbf{z}$, for each role r, we choose the best span s^*, such that $s^* = \operatorname{argmax}_{s \in \mathcal{S}_r} z_{r,s}$, solving ties arbitrarily.

4. **CPLEX**, *exact*: this tackles the actual ILP (Equation 5.8) with CPLEX.

5. **AD^3**, *LP*: this the counterpart of the LP version of CPLEX, where the relaxed problem is solved using AD^3. We choose the spans for each role in the same way as in strategy 3.

6. **AD^3**, *exact*: this couples AD^3 with branch-and-bound search to get the exact integer solution.

Table 5.9 shows performance of the different decoding strategies on the test set. We report precision, recall, and F_1 scores. For these experiments too, we use the evaluation script from SemEval 2007 shared task. Since these scores do not penalize structural violations, we also report the number of overlap, "excludes," and "requires" constraints that were violated in the test set. Finally, we tabulate each setting's decoding time in seconds on the whole test set averaged over 5 runs. We used a 64-bit machine with 2 2.6GHz dual-core CPUs (i.e., 4 processors in all) with a total of 8GB of RAM. The workers in AD^3 were not parallelized, while CPLEX automatically parallelized execution. The Local model is very fast but suffers degradation in precision and violates one constraint roughly per nine targets. The decoding strategy of §5.4.1 used a default beam size of 10,000, which is extremely slow; a faster version of beam size 100 results in the same precision and recall values, but is 15 times faster on our test set. Beam size 2 results in a bit worse precision and recall values, but is even faster. All of these, however, result in many constraint violations. Strategies involving CPLEX and AD^3 perform similarly to each other and to beam search on precision and recall, but eliminate most or all of the constraint violations. With respect to precision and recall, exact AD^3 and beam search with width 10000 were found to be statistically indistinguishable ($p > 0.01$). The decoding strategy with beam size 2 is 11–16 times faster than the CPLEX strategies, but is only twice as fast as AD^3, and results in significantly more structural violations. The exact algorithms are slower than the LP versions, but compared to CPLEX, AD^3 is significantly faster and has a narrower gap between its exact and LP versions. We found that relaxation was tight 99.8% of the time on the test examples.

The example in Fig. 5.1 is taken from our test set, and shows an instance where two roles, Partner_1 and Partner_2 share the "requires" relationship; for this example, the beam search decoder misses the Partner_2 role, which is a violation, while our AD^3 decoder identifies both arguments correctly. Note that beam search makes plenty of linguistic violations, but has precision and recall values that are marginally better than AD^3. We found that beam search, when violating many "requires" constraints, often finds one role in the pair, which increases its recall. AD^3 is sometimes more conservative in such cases, predicting neither role. Figure 5.3 shows such an example where beam search finds one role (Entity_1) while AD^3 is more conservative and predicts no roles. Figure 5.4 shows another example contrasting the output of beam search and AD^3 where the former predicts two roles sharing an "excludes" relationship; AD^3 does not violate this constraint and tries to predict a more consistent argument set. A second issue, as noted in footnote 28, is that the annotations sometimes violate these constraints. Figure 5.5 shows two examples where the FrameNet annotators do not respect the "requires" constraint; in figure 5.5(b), we observe a case where one role in a required pair appears in a previous sentence. Overall, we found it interesting that imposing the constraints did not have much effect on standard measures of accuracy.

ARGUMENT IDENTIFICATION				Violations				
Method	P	R	F_1	Overlap	Requires	Excludes	**Time in Secs.**	
Local	82.00	76.36	79.08	441	45	15	1.26	± 0.01
beam = 2	83.68	76.22	79.78	0	49	0	2.74	± 0.10
beam = 100	83.83	76.28	79.88	0	50	1	29.0	± 0.25
beam = 10000	83.83	76.28	79.88	0	50	1	440.67	± 5.53
CPLEX, *LP*	83.80	76.16	79.80	0	1	0	32.67	± 1.29
CPLEX, *exact*	83.78	76.17	79.79	0	0	0	43.12	± 1.26
AD^3, *LP*	83.77	76.17	79.79	2	2	0	4.17	± 0.01
AD^3, *exact*	83.78	76.17	79.79	0	0	0	4.78	± 0.04

Table 5.9: Comparison of decoding strategies in §5.5.4 on the dataset released with the **FrameNet 1.5 Release**, given *gold* frames. We evaluate in terms of precision, recall and F_1 score on our test set containing 4,458 targets. We also compute the number of structural violations each model makes: number of overlapping arguments and violations of the "requires" and "excludes" constraints of §5.5.2. Finally decoding time (without feature computation steps) on the *whole* test set is shown in the last column averaged over 5 runs.

5.6 Discussion

In this chapter, we have provided a supervised model for rich frame-semantic parsing, based on a combination of knowledge from FrameNet, two probabilistic models trained on full-text annotations released along with the FrameNet lexicon, and expedient heuristics. One of the models employs latent variables to model unseen lexical units in either the FrameNet lexicon or training data, and our results show that quite often, this model is able to find a closely related frame to the gold standard. The second model for argument identification, trained using maximum conditional log-likelihood, conjoins the two traditional steps of finding the potential arguments in a sentence and then labeling them as a role into one stage. Our system achieves improvements over the state of the art at each stage of processing and collectively. We compare our performance with the the state of the art on the SemEval 2007 benchmark dataset.

To further improve argument identification, we use a novel method for collectively predicting all arguments of a given target by incorporating declarative linguistic knowledge as constraints. It outperforms the naïve local decoding scheme that is oblivious to the constraints. Furthermore, it is significantly faster than a decoder employing a state-of-the-art proprietary solver; it is slower than beam search (our chosen decoding method for comparison with the state of the art), which is inexact and does not respect all linguistic constraints. Our method is easily amenable to the inclusion of more constraints, which would require minimal programming effort. Certain straightforward extensions of the parser, which we do not consider in this thesis are:

1. Addition of various other linguistic constraints based on information present in FrameNet, e.g. the semantic type of roles and constraints over frame disambiguation across targets in a document.

2. We observe, as in Figure 5.5, that there are instances where the annotators do not respect the "requires" constraint. We treat this constraint as a hard factor in our inference scheme; an extension would convert this factor into one that incorporates a "soft" constraint, to be learned from annotated data.

3. We note that syntactic parse errors lead to poor selection of potential arguments; domain adaptation of a syntactic parser, using strategies presented by Hall et al. (2011) could potentially serve as a better preprocessor for frame-semantic parsing.

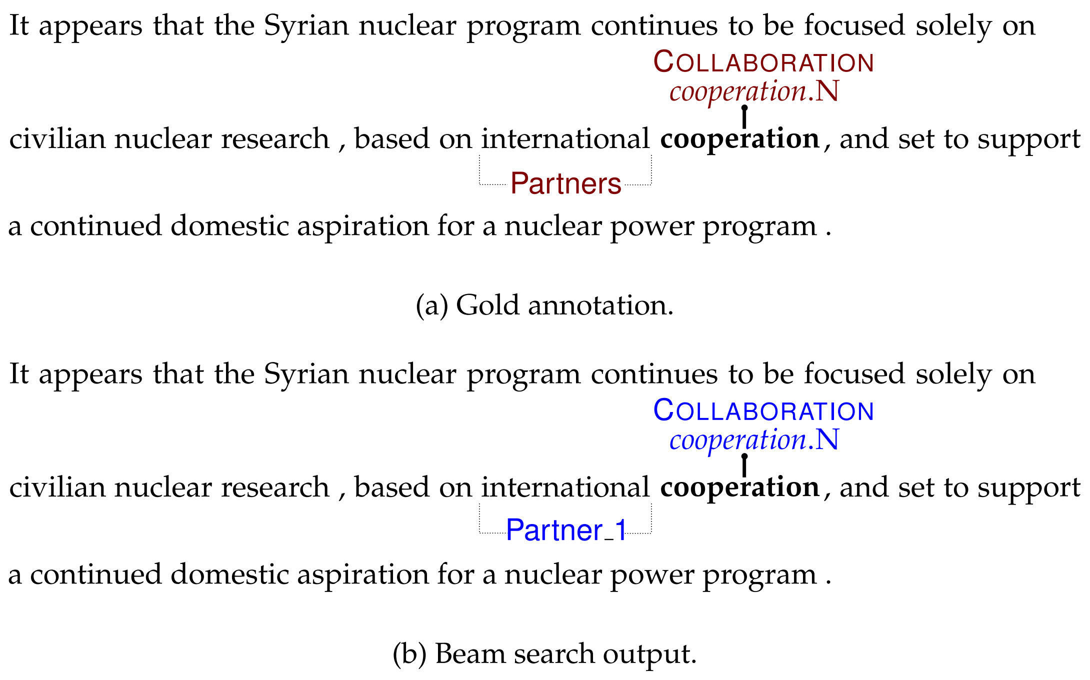
There are a few long term future directions that one could take. Here, we presented a joint model for finding all the arguments collectively, given the disambiguated frame for a chosen predicate. An extension would be to perform frame and argument identification within a joint inference scheme, without solving these two subtasks in a pipeline.

We made a naïve attempt at jointly disambiguating frames and their corresponding arguments with separately trained frame and argument identification models, without any scaling strategy that would treat the contribution of these two models differently.[29] Results with this joint inference scheme were statistically indistinguishable from the results presented in this chapter for both frame identification and full frame-semantic parsing; however, inference was much slower than the pipeline. We leave thorough exploration of joint frame and argument identification learning and inference to future work.

Another extension of our frame-semantic parsing model would be the inclusion of more linguistically motivated features and constraints. Although the current set of features in the argument identification model looks at the subcategorization frame of the target in question, it could be improved. For example, we could impose constraints on the set of arguments that a target could take; an example constraint for a verbal target that is ditransitive would be the requirement of two arguments whose grammatical functions are two objects in the syntactic parse. Such constraints, if available, can be easily placed using the ILP framework we presented in §5.5.2. The VerbNet lexicon (Schuler, 2005) contains such information regarding verbs; future work could relate the information in VerbNet and map them to the FrameNet lexicon and use it to impose further constraints on argument identification inference.

In the next chapter, we observe how distributional similarity can be used as to improve the coverage of the frame-semantic parser, by probabilistically expanding our lexicon. To this end, we use graph-based semi-supervised learning to learn possible semantic frames on targets unseen in FrameNet or any supervised data. The parser described in this chapter is available for download at `http://www.ark.cs.cmu.edu/SEMAFOR`.

[29]Related research in shallow semantic parsing have explored ideas with scaling strategies that treat the contribution of various separately trained systems differently; for an example, see Srikumar and Roth (2011).

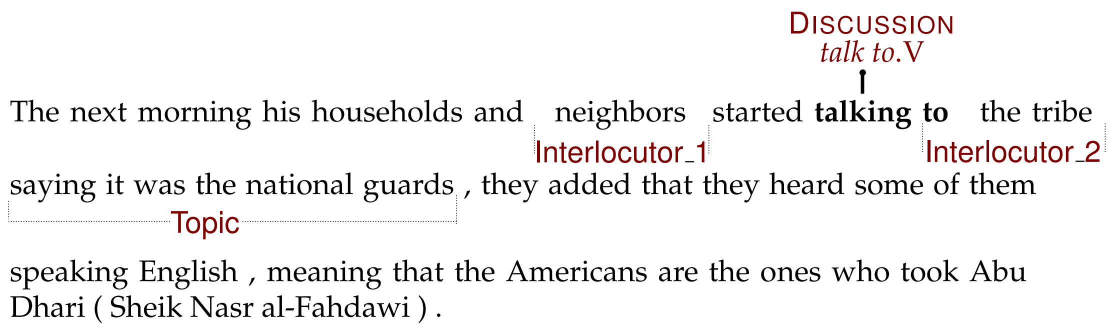

It appears that the Syrian nuclear program continues to be focused solely on civilian nuclear research , based on international **cooperation** , and set to support a continued domestic aspiration for a nuclear power program .

(a) Gold annotation.

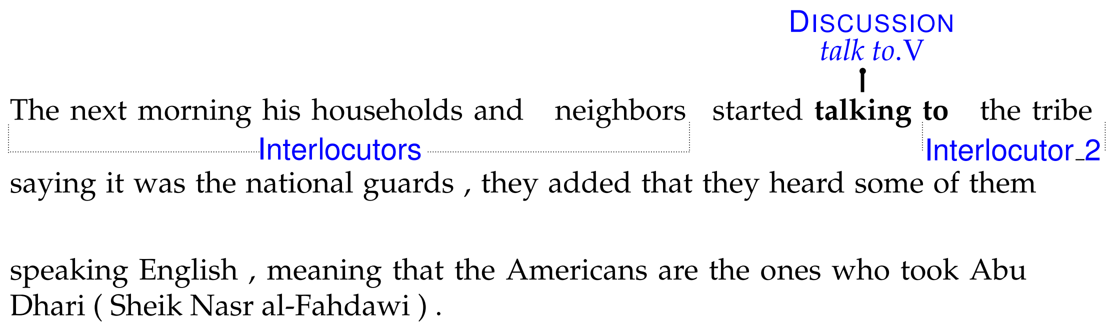
It appears that the Syrian nuclear program continues to be focused solely on civilian nuclear research , based on international **cooperation** , and set to support a continued domestic aspiration for a nuclear power program .

(b) Beam search output.

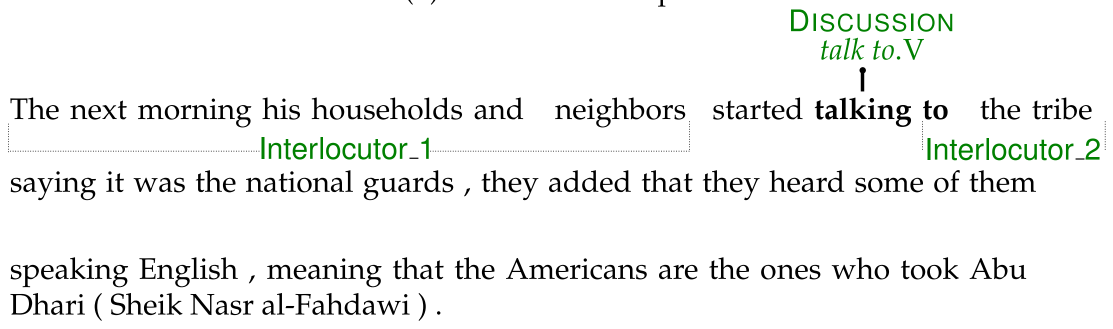
Figure 5.3: An example from the test set where (a) exhibits the gold annotation for a target that evokes the COLLABORATION frame, with the Partners role fulfilled by the span "international". (b) shows the prediction made by the beam search decoding scheme (beam = 10000), where it marks "international" with the Partners_1 role, which violates the "requires" constraint; FrameNet notes that this role should be present with the Partners_2 role. AD^3 proves to be conservative and predicts no role – it is penalized by the evaluation script, but does not produce output that violates linguistic constraints.


DISCUSSION
talk to.V
The next morning his households and neighbors started **talking to** the tribe
Interlocutor_1
Interlocutor_2
saying it was the national guards , they added that they heard some of them
Topic
speaking English , meaning that the Americans are the ones who took Abu Dhari (Sheik Nasr al-Fahdawi) .

(a) Gold annotation.

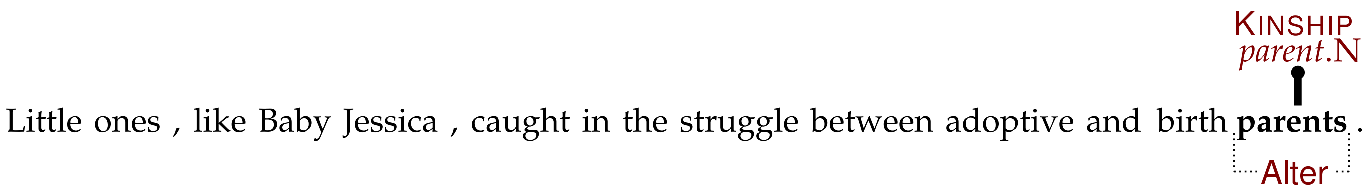

DISCUSSION
talk to.V
The next morning his households and neighbors started **talking to** the tribe
Interlocutors
Interlocutor_2
saying it was the national guards , they added that they heard some of them
speaking English , meaning that the Americans are the ones who took Abu Dhari (Sheik Nasr al-Fahdawi) .

(b) Beam search output.

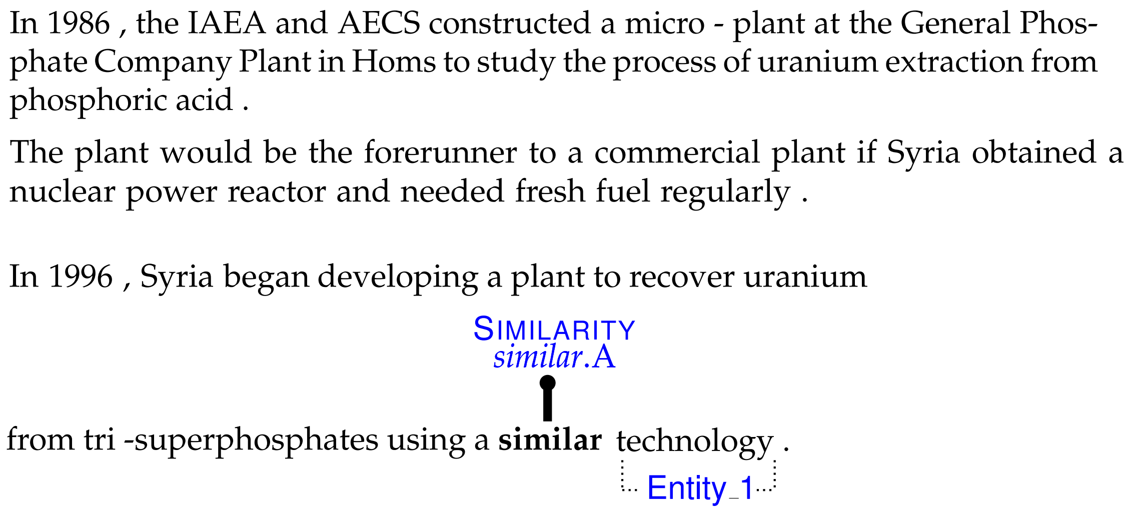

DISCUSSION
talk to.V
The next morning his households and neighbors started **talking to** the tribe
Interlocutor_1
Interlocutor_2
saying it was the national guards , they added that they heard some of them
speaking English , meaning that the Americans are the ones who took Abu Dhari (Sheik Nasr al-Fahdawi) .

(c) AD^3 output.

Figure 5.4: An example from the test set where (a) exhibits the gold annotation for a target that evokes the DISCUSSION frame, with the Interlocutor_1 role fulfilled by the span "neighbors." (b) shows the prediction made by the beam search decoding scheme (beam = 10000), where it marks "The next morning his households and neighbors" with the Interlocutors role, which violates the "excludes" constraint with respect to the Interlocutor_2 role. In (c), AD^3 marks the wrong span as the Interlocutor_1 role, but it does not violate the constraint. Both beam and AD^3 inference miss the Topic role.


Little ones , like Baby Jessica , caught in the struggle between adoptive and birth **parents** .

(a) Gold annotation.


In 1986 , the IAEA and AECS constructed a micro - plant at the General Phosphate Company Plant in Homs to study the process of uranium extraction from phosphoric acid .

The plant would be the forerunner to a commercial plant if Syria obtained a nuclear power reactor and needed fresh fuel regularly .

In 1996 , Syria began developing a plant to recover uranium from tri -superphosphates using a **similar** technology .

(b) Gold annotation.

Figure 5.5: In (a), the annotators only annotate the Alter role, while the lexicon mentions that this role, if present warrants the presence of the Ego role. (b) shows three sentences, with the last sentence showing an instance of the SIMILARITY frame. In this example, only Entity_1 is marked by the annotators, because the other required role Entity_2 is absent in the current sentence, but can be traced back to a previous sentence.

Chapter 6

Semi-Supervised Lexicon Expansion

In this chapter, we will investigate a generic semi-supervised learning approach towards expanding natural language *type* lexicons. Type lexicons are widely used in various tasks in language processing. These lexicons contain natural language types that appear as token instances in free text, containing the potential set of labels that they associate with. Within the context of this thesis, the set of lexical unit types along with the potential set of semantic frames they evoke is an example of a type lexicon (Chapter 5). Another example is the collection of words in a language, with each word paired with the potential set of POS tags that they can associate with. POS lexicons have been used in unsupervised POS tagging experiments (Merialdo, 1994; Smith and Eisner, 2005; Das and Petrov, 2011). A gazetteer used for named-entity recognition is another example.

Here, we will focus on some generic techniques that expand a seed lexicon into a noisier, but much larger set of types with a probability distribution over labels on each of them. We will also observe how such an expanded lexicon can be used to constrain inference, especially keeping in mind the frame-semantic parsing task at hand. The research described in this chapter has been presented earlier in two conference papers (Das and Smith, 2011, 2012).

Semi-supervised learning is attractive for the learning of complex phenomena, for example, linguistic structure, where data annotation is expensive. Natural language processing applications have benefited from various SSL techniques, such as distributional word representations (Huang and Yates, 2009; Turian et al., 2010; Dhillon et al., 2011), self-training (McClosky et al., 2006), and entropy regularization (Jiao et al., 2006; Smith and Eisner, 2007). Here, we focus on semi-supervised learning that uses a *graph* constructed from labeled and unlabeled data. This framework, graph-based SSL—see Bengio et al. (2006) and Zhu (2008) for introductory material on this topic—has been widely used and has been shown to perform better than several other semi-supervised algorithms on benchmark datasets (Chapelle et al., 2006, ch. 21). The method constructs a graph where a small portion of vertices correspond to labeled instances, and the rest are unlabeled. Pairs of vertices are connected by weighted edges denoting the similarity between the pair. Traditionally, Markov random walks (Szummer and Jaakkola, 2001; Baluja et al., 2008) or optimization of a loss function based on smoothness properties of the graph (Corduneanu and Jaakkola, 2003; Zhu et al., 2003; Subramanya and Bilmes, 2008, *inter alia*) are performed to propagate labels from the labeled vertices to the unlabeled ones.

In this work, we are interested in multi-class generalizations of graph-propagation algorithms, where each graph vertex can assume one *or more* out of many possible labels (Talukdar and Crammer, 2009; Subramanya and Bilmes, 2008, 2009). For us, graph vertices correspond to natural language types (not tokens) and undirected edges between them are weighted using a similarity metric. Recently, this setup has been used to learn soft labels on natural language types (say, word n-grams or syntactically disambiguated predicates) to constrain structured prediction models. Applications have ranged from domain adaptation of part-of-speech (POS) taggers (Subramanya et al., 2010) to unsupervised learning of POS taggers by using bilingual graph-based projections (Das and Petrov, 2011). However, none of these approaches captured the empirical fact that only a few categories are actually possible for a given type (vertex). Take the case of POS tagging: Subramanya et al. (2010) construct a graph over trigram types as vertices, with 45 possible tags for the middle word of a trigram as the label set for each vertex. It is empirically observed that contextualized word types usually assume very few (most often, one) POS tags. However, along with graph smoothness terms, they apply a penalty that encourages distributions to be close to uniform, the premise being that it would maximize the entropy of the distribution for a vertex that is far away or disconnected from a labeled vertex (Subramanya et al., 2010, see Equation 2). To prefer maximum entropy solutions in low confidence regions of graphs, a similar entropic penalty is applied by (Subramanya and Bilmes, 2008, 2009).

Here, we make two major algorithmic contributions. First, we relax the assumption made by most previous work (Baluja et al., 2008; Das and Petrov, 2011; Subramanya and Bilmes, 2008, 2009; Subramanya et al., 2010; Zhu and Ghahramani, 2002) that the ℓ_1 norm of the scores assigned to the labels for a given vertex must be 1. In other words, in our framework, the label distribution at each vertex is *unnormalized*—the only constraint we put on the vertices' vectors is that they must be nonnegative. Moreover, we also assume the edge weights in a given graph are unconstrained, consistent with prior work on graph-based SSL (Das and Petrov, 2011; Subramanya and Bilmes, 2008, 2009; Subramanya et al., 2010; Zhu and Ghahramani, 2002). This relaxation lets us experiment with complex graph smoothness terms, including one that uses the symmetric entropic Jensen-Shannon divergence (Burbea and Rao, 1982; Lin, 1991), and also lets us use penalties that prefer vertex-level sparsity.

Second, we replace the penalties that prefer maximum entropy, used in prior work, with penalties that aim to identify *sparse* unnormalized measures at each graph vertex. We achieve this goal by penalizing the graph propagation objective with the ℓ_1 norm or the mixed $\ell_{1,2}$ norm (Kowalski and Torrésani, 2009) of the measures at each vertex, aiming for global and vertex-level sparsity, respectively.

The combination of the above two properties of our framework retains several attractive properties of prior work. Most importantly, the proposed graph objective functions are convex, so we avoid degenerate solutions and local minima. Moreover, the only set of linear constraints applied on the objectives require that the mass assigned to each label for each vertex be nonnegative. The relaxation of simplex constraints present in prior work lets us use a generic straightforward quasi-Newton method for optimization (Zhu et al., 1997).

In this chapter, we present experiments on two lexicon expansion problems in a semi-supervised setting:

1. Inducing distributions of POS tags over n-gram types in the *Wall Street Journal* section of the Penn Treebank corpus (Marcus et al., 1993); this set of experiments diverges from the dissertation's theme of learning for natural language semantics, but it shows the general efficacy of using our family of graph objectives.

2. Inducing distributions of semantic frames over lexical units unseen in annotated data.

Our methods produce sparse measures at graph vertices resulting in compact lexicons, and also result in better predictive performance with respect to the multi-class generalization of label propagation using Gaussian penalties (Zhu and Ghahramani, 2002) and entropic measure propagation (Subramanya and Bilmes, 2009), two state-of-the-art graph propagation algorithms. The chapter is organized as follows. §6.1 presents the statistical model we use for graph-based SSL. We present a family of graph objectives that make optimization easier than prior work and also employs sparse penalties. §6.2 presents a short discussion on how we optimize our objective functions, followed by two sections that focuses on the experiments conducted and the results achieved (§6.3-6.4). Finally, we conclude this chapter in §6.5.

6.1 Model

6.1.1 Graph-Based SSL as Inference in a Markov Network

Let $\mathcal{D}_l = \{(\mathbf{x}_j, \hat{q}_j)\}_{j=1}^{l}$ denote l annotated data *types*;[1] $\mathbf{x}_j$'s empirical label distribution is $\hat{q}_j$. Let the unlabeled data types be denoted by $\mathcal{D}_u = \{\mathbf{x}_i\}_{i=l+1}^{m}$. Usually, $l \ll m$.

[1] As it will become clearer with further exposition in §6.3-6.4, these types are entities like n-grams or individual lexical units, not tokens in running text.

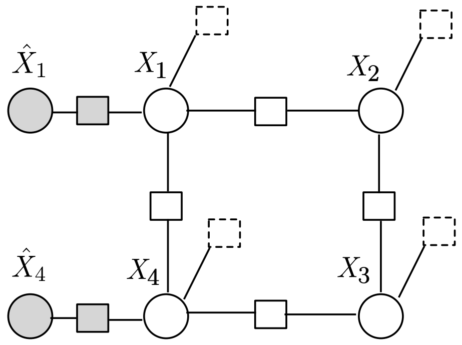


Figure 6.1: An example factor graph for the graph-based SSL problem. See text for the significance of the shaded and dotted factors, and the shaded variables.

Thus, the entire dataset can be called $\mathcal{D} \triangleq \{\mathcal{D}_l, \mathcal{D}_u\}$. Traditionally, the graph-based SSL problem has been set up as follows. Let $\mathcal{G} = (V, E)$ correspond to an undirected graph with vertices V and edges E. $\mathcal{G}$ is constructed by transforming each data type $\mathbf{x}_i \in \mathcal{D}$ to a vertex; thus $V = \{1, 2, \ldots, m\}$, and $E \subseteq V \times V$. Let V_l (V_u) denote the labeled (unlabeled) vertices. Moreover, we assume a symmetric weight matrix $\mathbf{W}$ that defines the similarity between a pair of vertices $i, k \in V$. We first define a component of this matrix as $w_{ij} \triangleq [\mathbf{W}]_{ik} = \mathrm{sim}(\mathbf{x}_i, \mathbf{x}_k)$. We also fix $w_{ii} = 0$ and set $w_{ik} = w_{ki} = 0$ if $k \notin \mathcal{N}(i)$ and $i \notin \mathcal{N}(k)$, where $\mathcal{N}(j)$ denotes the K-nearest neighbors of vertex j. The last modification makes the graph sparse.[2] We next define an unnormalized measure q_i for every vertex $i \in V$. As mentioned before, we have $\hat{q}_j$, a probability distribution estimated from annotated data for a labeled vertex $j \in V_l$. q_i and $\hat{q}_j$ are $|Y|$-dimensional measures, where Y is the possible set of labels; while $\hat{q}_j$ lies within the $|Y|$-dimensional probability simplex, q_i are unnormalized with each component $q_i(y) \geq 0$. For some applications, $\hat{q}_j$ are expected to be sparse, usually with only one or two components active, the rest being zero.

Graph-based SSL aims at finding the best $\boldsymbol{q} = \{q_i : 1 \leq i \leq m\}$ given the supervised empirical distributions $\hat{q}_j$, and the weight matrix $\mathbf{W}$, which provides the geometry of all the vertices. We represent this problem using a pairwise Markov network (MN). For every vertex (including labeled ones) $i \in V$, we create a variable X_i. Additionally, for labeled vertices $j \in V_l$, we create variables $\hat{X}_j$. All variables in the MN are defined to be *vector-valued*; specifically, variables X_i, $\forall i \in V$, take value q_i, and variables $\hat{X}_j$ corresponding to the labeled vertices in $\mathcal{G}$ are observed with values $\hat{q}_j$. An example

[2]This is not the same kind of sparsity that our novel approach seeks.

factor graph for this MN, with only four vertices, is shown in Figure 6.1. In the figure, the variables indexed by 1 and 4 correspond to labeled vertices. Factor ϕ_j with scope $\{X_j, \hat{X}_j\}$ encourages q_j to be close to $\hat{q}_j$. For every edge $i-k \in E$, factor φ_{i-k} encourages similarity between q_i and q_k, making use of the weight matrix $\mathbf{W}$ (i.e., when w_{ik} is larger, the two measures are more strongly encouraged to be close). These factors are white squares with solid boundaries in the figure. Finally, we define unary factors on all variables $X_i, i \in V$, named $\psi_i(X_i)$, that can incorporate prior information. In Figure 6.1, these factors are represented by white squares with dashed boundaries.

According to the factor graph, the joint probability for all the measures q_i, $\forall i \in V$ that we want to induce, is defined as:

$$P(\boldsymbol{X}; \Phi) = \frac{1}{Z} \prod_{j=1}^{l} \phi_j(X_j, \hat{X}_j) \cdot \prod_{i-k \in E} \varphi_{i-k}(X_i, X_k) \cdot \prod_{i=1}^{m} \psi_i(X_i)$$

where Φ is the set of all factors in the factor graph, and Z is a partition function that normalizes the factor products for a given configuration of $\boldsymbol{q}$. Since the graph-based SSL problem aims at finding the best $\boldsymbol{q}$, we optimize $\ln P(\boldsymbol{X}; \Phi)$; equivalently,

$$\underset{\boldsymbol{q} \text{ s.t } \boldsymbol{q} \geq \mathbf{0}}{\operatorname{argmax}} \sum_{j=1}^{l} \ln \phi_j(X_j, \hat{X}_j) + \sum_{i-k \in E} \ln \varphi_{i-k}(X_i, X_k) + \sum_{i=1}^{m} \ln \psi_i(X_i) \tag{6.1}$$

The above is an optimization problem with only non-negativity constraints. It equates to maximum *a posteriori* (MAP) inference; hence, the partition function Z can be ignored. We next discuss the nature of the three different factors in Equation 6.1.

6.1.2 Log-Factors as Penalties

The nature of the three types of factors in Equation 6.1 governs the behavior of a graph-based SSL algorithm. Hence, the equation specifies a *family* of graph-based methods that generalize prior research. We desire the following properties to be satisfied in the factors:

1. Convexity of Equation 6.1.

2. Amenability to high-performance optimization algorithms without extra hyper-parameters and with the option of parallelization.

3. Sparse solutions as expected in natural language lexicons.

Pairwise factors

In our work, for the pairwise factors $\phi_j(X_j, \hat{X}_j)$ and $\varphi_{i-k}(X_i, X_k)$, we examine two types of distances that attempt to capture inconsistencies between neighboring vertices: the squared ℓ_2 norm and the Jensen-Shannon (JS) divergence (Burbea and Rao, 1982; Lin, 1991), which is a symmetrized generalization of the Kullback-Leibler (KL) divergence (Kullback and Leibler, 1951; Cover and Thomas, 1991). These two divergences are symmetric and are inspired by previous work; however, the use of the JS divergence is novel and extends the work of Subramanya and Bilmes (2008), who used the asymmetric KL divergence (with the simplex constraints on each vertex) as pairwise penalties due to the ease of optimization. Specifically, the factors look like:

$$\begin{aligned} \ln \phi_j(X_j, \hat{X}_j) &= -\delta(q_j, \hat{q}_j) && (6.2) \\ \ln \varphi_{i-k}(X_i, X_k) &= -2 \cdot \mu \cdot w_{ik} \cdot \delta(q_i, q_k) && (6.3) \end{aligned}$$

where μ is a hyperparameter whose choice we discuss in §6.3-6.4. The function $\delta(u, v)$ for two vectors u and v are defined in two ways:

$$\begin{aligned} \underset{\text{Gaussian}}{\delta(u, v)} &= \|u - v\|_2^2 && (6.4) \\ \underset{\text{Entropic}}{\delta(u, v)} &= \frac{1}{2} \sum_{y \in Y} \left(u(y) \cdot \ln \frac{2 \cdot u(y)}{u(y) + v(y)} + v(y) \cdot \ln \frac{2 \cdot v(y)}{u(y) + v(y)} \right) && (6.5) \end{aligned}$$

We call the version of $\delta(u, v)$ that uses the squared ℓ_2 distance (Equation 6.4) *Gaussian*, as it equates to label propagation via Gaussian fields proposed by Zhu et al. (2003). A minor difference lies in the fact that we include variables $X_j, j \in V_l$ for labeled vertices too, and allow them to change, but penalize them if they go too far away from the observed labeled distributions $\hat{q}_j$. The other $\delta(u, v)$ shown in Equation 6.5 uses the generalized JS-divergence defined in terms of the generalized KL-divergence over unnormalized measures (O'Sullivan, 1998):

$$D_{KL}(u \| v) = \sum_y \left(u(y) \ln \frac{u(y)}{v(y)} - u(y) + v(y) \right) \qquad (6.6)$$

Using the definition of the generalized KL divergence in Equation 6.6, we get the generalized version of the JS-divergence as:

$$\begin{aligned}
D_{JS}(u \parallel v) &= \frac{1}{2} D_{KL}\left(u \parallel \frac{u+v}{2}\right) + \frac{1}{2} D_{KL}\left(v \parallel \frac{u+v}{2}\right) \\
&= \frac{1}{2} \sum_{y \in Y} \Bigg(u(y) \cdot \ln \frac{2 \cdot u(y)}{u(y)+v(y)} - u(y) + \frac{1}{2}\big(u(y)+v(y)\big) \\
&\qquad + v(y) \cdot \ln \frac{2 \cdot v(y)}{u(y)+v(y)} - v(y) + \frac{1}{2}\big(u(y)+v(y)\big)\Bigg) \\
&= \frac{1}{2} \sum_{y \in Y} \left(u(y) \cdot \ln \frac{2 \cdot u(y)}{u(y)+v(y)} + v(y) \cdot \ln \frac{2 \cdot v(y)}{u(y)+v(y)} \right) \qquad (6.7)
\end{aligned}$$

This form for the generalized JS-divergence ends up looking exactly similar to the traditional version using normalized probability distributions. Equation 6.5 improves prior work by replacing the asymmetric KL-divergence used to bring the distributions at labeled vertices close to the corresponding observed distributions, as well as replacing the KL-based graph smoothness term with the symmetric JS-divergence (Subramanya and Bilmes, 2008, see first two terms in Equation 1). Empirical evidence shows that entropic divergences help in multiclass problems where a vertex can assume multiple labels, and may perform better than objectives with quadratic penalties (Subramanya and Bilmes, 2008, 2009).

A major departure from prior work is the use of unnormalized measures in Equation 6.4-6.5, which simplifies optimization even with the complex JS-divergence in the objective function (see §6.2), and, we will see, produces comparable and often better results than baselines using normalized distributions (see §6.3-6.4).

Unary factors

The unary factors in our factor graph $\psi_i(X_i)$ can incorporate prior information specific to a particular vertex $\mathbf{x}_i$ embodied by the variable X_i. Herein, we examine three straightforward penalties, which can be thought of as penalties that encourage either uniformity or sparsity:

$$\text{Uniform squared } \ell_2\text{:} \quad \ln \psi_i(X_i) = -\lambda \cdot \left\| q_i - \tfrac{1}{|Y|} \right\|_2^2 \qquad (6.8)$$

$$\text{Sparse } \ell_1\text{:} \quad \ln \psi_i(X_i) = -\lambda \cdot \| q_i \|_1 \qquad (6.9)$$

$$\text{Sparse } \ell_{1,2}\text{:} \quad \ln \psi_i(X_i) = -\lambda \cdot \| q_i \|_1^2 \qquad (6.10)$$

where λ is a hyperparameter whose choice we discuss in §6.3-6.4. The first penalty expressed in Equation 6.8 penalizes q_i if it is far away from the uniform distribution. This penalty has been used previously (Das and Petrov, 2011; Subramanya et al., 2010), and is similar to the maximum entropy penalty of (Subramanya and Bilmes, 2008, 2009). The intuition behind its use is that for low confidence or disconnected regions, one would prefer to have a uniform measure on a graph vertex. The penalties in equations 6.9–6.10, on the other hand, encourage sparsity in the measure q_i; these are related to regularizers for generalized linear models: the lasso (Tibshirani, 1996) and the elitist lasso (Kowalski and Torrésani, 2009). The former encourages global sparsity, the latter sparsity per vertex.[3] The $\ell_{1,2}$ penalty is calculated as:

$$\|q_i\|_1^2 = (\sum_{y \in Y} |q_i(y)|^1)^2 \tag{6.11}$$

The $\ell_{1,2}$ penalty aims at sparsity per vertex because the ℓ_1 norm of its components prefers sparsity within the vertex; next, once we take the squared ℓ_2 norm of the ℓ_1 norms spanning all vertices in the graph, it promotes density across the ℓ_1 norms.

The latter two sparsity inducing penalties are desirable for natural language type learning. For natural language applications of graph-based SSL, especially in low-resource scenarios with very few labeled datapoints, one would like to enforce the fact that unlabeled datapoints can have very few labels. Talukdar (2010) enforced label sparsity for information extraction by discarding labels with low scores during graph propagation updates, but did not use a principled mechanism to arrive at sparse measures at graph vertices. Unlike the uniform penalty (equation 6.8), sparsity corresponds to the idea of entropy *minimization* (Grandvalet and Bengio, 2004). Since we use unnormalized measures at each variable X_i, for low confidence graph regions or disconnected vertices, sparse penalties will result in all zero components in q_i, which conveys that the graph propagation algorithm is not confident on any potential label, a condition that is perfectly acceptable.

Model variants

We compare six objective functions: we combine factor representations from each of equations 6.4–6.5 with those from each of equations 6.8–6.10, replacing them in the generic graph objective function of Equation 6.1. The nature of these six models is succinctly summarized in Table 6.1. We note their corresponding objective functions below:

[3] One could additionally experiment with a non-sparse penalty based on the squared ℓ_2 norm with zero mean: $\ln \psi_i(X_i) = -\lambda \cdot \|q_i\|_2^2$. We experimented with this unary penalty (along with the pairwise Gaussian penalty for binary factors) for the semantic frame lexicon expansion problem, and found that it performs exactly at par with the squared ℓ_2 penalty with uniform mean. To limit the number of non-sparse graph objectives, we omit additional experiments with this unary penalty.

Abbrev.	Factors	
	Pairwise	Unary
UGF-ℓ_2	Gaussian	Uniform squared ℓ_2
UGF-ℓ_1	Gaussian	Sparse ℓ_1
UGF-$\ell_{1,2}$	Gaussian	Sparse $\ell_{1,2}$
UJSF-ℓ_2	Entropic	Uniform squared ℓ_2
UJSF-ℓ_1	Entropic	Sparse ℓ_1
UJSF-$\ell_{1,2}$	Entropic	Sparse $\ell_{1,2}$

Table 6.1: Six variants of graph objective functions novel to this work. These variants combine the pairwise factor representations from equations 6.4–6.5 with unary factor representations from each of equations 6.8–6.10 (which either encourage uniform or sparse measures), to be used in the graph objective function expressed in Equation 6.1.

1. Unnormalized Gaussian fields with a squared ℓ_2 penalty (**UGF-ℓ_2**):

$$\underset{\boldsymbol{q},\text{ s.t. }\boldsymbol{q}\geq\boldsymbol{0}}{\arg\min} \sum_{j=1}^{l} \|q_j - \hat{q}_j\|_2^2 + \sum_{i=1}^{m}\left(\mu \sum_{k\in\mathcal{N}(i)} w_{ik}\,\|q_i - q_k\|_2^2 + \lambda \left\|q_i - \tfrac{1}{|Y|}\right\|_2^2\right) \tag{6.12}$$

2. Unnormalized Gaussian fields with an ℓ_1 penalty (**UGF-ℓ_1**):

$$\underset{\boldsymbol{q},\text{ s.t. }\boldsymbol{q}\geq\boldsymbol{0}}{\arg\min} \sum_{j=1}^{l} \|q_j - \hat{q}_j\|_2^2 + \sum_{i=1}^{m}\left(\mu \sum_{k\in\mathcal{N}(i)} w_{ik}\,\|q_i - q_k\|_2^2 + \lambda \|q_i\|_1\right) \tag{6.13}$$

3. Unnormalized Gaussian fields with an $\ell_{1,2}$ penalty (**UGF-$\ell_{1,2}$**):

$$\underset{\boldsymbol{q},\text{ s.t. }\boldsymbol{q}\geq\boldsymbol{0}}{\arg\min} \sum_{j=1}^{l} \|q_j - \hat{q}_j\|_2^2 + \sum_{i=1}^{m}\left(\mu \sum_{k\in\mathcal{N}(i)} w_{ik}\,\|q_i - q_k\|_2^2 + \lambda \|q_i\|_1^2\right) \tag{6.14}$$

4. Unnormalized Jensen-Shannon fields with a squared ℓ_2 penalty (**UJSF-ℓ_2**):

$$\underset{\boldsymbol{q},\text{ s.t. }\boldsymbol{q}\geq\boldsymbol{0}}{\arg\min} \sum_{j=1}^{l} D_{JS}(q_j \parallel \hat{q}_j) + \sum_{i=1}^{m}\left(\mu \sum_{k\in\mathcal{N}(i)} w_{ik} D_{JS}(q_i \parallel q_k) + \lambda \left\|q_i - \tfrac{1}{|Y|}\right\|_2^2\right) \tag{6.15}$$

5. Unnormalized Jensen-Shannon fields with an ℓ_1 penalty (**UJSF-ℓ_1**):

$$\underset{\boldsymbol{q},\text{ s.t. }\boldsymbol{q}\geq\mathbf{0}}{\arg\min} \sum_{j=1}^{l} D_{JS}(q_j \parallel \hat{q}_j) + \sum_{i=1}^{m}\left(\mu \sum_{k\in\mathcal{N}(i)} w_{ik} D_{JS}(q_i \parallel q_k) + \lambda \left\|q_i\right\|_1\right) \tag{6.16}$$

6. Unnormalized Jensen-Shannon fields with an $\ell_{1,2}$ penalty (**UJSF-$\ell_{1,2}$**):

$$\underset{\boldsymbol{q},\text{ s.t. }\boldsymbol{q}\geq\mathbf{0}}{\arg\min} \sum_{j=1}^{l} D_{JS}(q_j \parallel \hat{q}_j) + \sum_{i=1}^{m}\left(\mu \sum_{k\in\mathcal{N}(i)} w_{ik} D_{JS}(q_i \parallel q_k) + \lambda \left\|q_i\right\|_1^2\right) \tag{6.17}$$

For each model, we find the best set of measures $\boldsymbol{q}$ that minimize the corresponding graph objective functions, such that $\boldsymbol{q} \geq \mathbf{0}$. Note that in each of the graph objectives, we have two hyperparameters μ and λ that control the influence of the second and the third terms of Equation 6.1 respectively. We discuss how these hyperparameters are chosen in §6.3-6.4.

Baseline Models

We compare the performance of the six graph objectives of Table 6.1 with two strong baselines that have been used in previous work. These two models use the following two objective functions, and find $\boldsymbol{q}$ s.t. $\boldsymbol{q} \geq \mathbf{0}$ and $\forall i \in V, \sum_{y\in Y} q_i(y) = 1$. The first is a normalized Gaussian field with a squared uniform ℓ_2 penalty as the unary factor (**NGF-ℓ_2**):

$$\underset{\substack{\boldsymbol{q},\text{ s.t. }\boldsymbol{q}\geq\mathbf{0},\\ \forall i\in V, \|q_i\|_1=1}}{\arg\min} \sum_{j=1}^{l} \|q_j - \hat{q}_j\|_2^2 + \sum_{i=1}^{m} (\mu \sum_{k\in\mathcal{N}(i)} w_{ik} \|q_i - q_k\|_2^2 + \lambda \left\|q_i - \tfrac{1}{|Y|}\right\|_2^2) \tag{6.18}$$

The second is a normalized KL field with an entropy penalty as the unary factor (**NKLF-ME**):

$$\underset{\substack{\boldsymbol{q},\text{ s.t. }\boldsymbol{q}\geq\mathbf{0},\\ \forall i\in V, \|q_i\|_1=1}}{\arg\min} \sum_{j=1}^{l} D_{KL}(\hat{q}_j \parallel q_j) + \sum_{i=1}^{m} (\mu \sum_{k\in\mathcal{N}(i)} w_{ik} D_{KL}(q_i \parallel q_k) - \lambda \cdot H(q_i)) \tag{6.19}$$

where $H(q_i)$ denotes the Shannon entropy of the distribution q_i. Both these objectives are constrained by the fact that every q_i must be within the $|Y|$-dimensional probability simplex. The objective function in 6.18 has been used previously (Subramanya et al.,

2010) and serves as a generalization of Zhu et al. (2003). The entropic objective function in 6.19, originally called *measure propagation*, performed better at multiclass problems when compared to graph objectives using the quadratic criterion (Subramanya and Bilmes, 2008).

6.2 Optimization

The six variants of Equation 6.1 in Table 6.1 (corresponding to Equations 6.12-6.17) are convex. This is because the ℓ_1, squared ℓ_2 and the $\ell_{1,2}$ penalties are convex. Moreover, the generalized JS-divergence term, which is a sum of two KL-divergence terms, is convex (Cover and Thomas, 1991). Since we choose μ, λ and w_{ik} to be nonnegative, these terms' sums are also convex in $q_i, i \in V$. The graph objectives of the two baselines noted in expressions 6.18–6.19 are also convex because negative entropy in expression 6.19 is convex, and rest of the penalties are the same as our six objectives.

In our work, to optimize the objectives of Table 6.1, we use a generic quasi-Newton gradient-based optimizer that can handle bound-inequality constraints, called L-BFGS-B (Zhu et al., 1997). Partial derivatives of the graph objectives are computed with respect to each parameter $\forall i, y, q_i(y)$ of $\boldsymbol{q}$ and passed on to the optimizer which updates them such that the objective function of Equation 6.1 is maximized. Note that since the ℓ_1 and $\ell_{1,2}$ penalties are non-differentiable at 0, special techniques are usually used to compute updates for unconstrained parameters (Andrew and Gao, 2007). However, since $\boldsymbol{q} \geq \mathbf{0}$, their absolute value can be assumed to be right-continuous, making the function differentiable. Thus,

$$\frac{\partial}{\partial q_i(y)} \|q_i\|_1 = 1 \qquad\qquad \frac{\partial}{\partial q_i(y)} \|q_i\|_1^2 = 2 \cdot \|q_i\|_1$$

We omit the form of the derivatives of the other penalties, as they are straightforward to calculate. There are several advantages to taking this route towards optimization. The ℓ_2 and the JS-divergence penalties for the pairwise terms can be replaced with more interesting convex divergences if required, and still optimization will be straightforward. Moreover, the nonnegative constraints make optimization with sparsity inducing penalties easy. Finally, computing the objective function and the partial derivatives is easily parallelizable on multi-core (Gropp et al., 1994) or cluster (Dean and Ghemawat, 2008) architectures, by dividing up the computation across graph vertices, using MapReduce (see §3.3), a route that we adopt in this work. In comparison, constrained problems such as the one in Equation 6.19 require a specialized alternating minimization technique (Subramanya and Bilmes, 2008, 2009), that performs two passes through the graph vertices during one iteration of updates, introduces an auxiliary set of probability distributions at each graph vertex (thus, increasing memory requirements) and another hyperparameter that is used to transform the weight matrix $\mathbf{W}$ to be suitable for the alternating minimization procedure. To optimize the baseline objectives, we borrow the gradient-free iterative updates described by Subramanya and Bilmes (2009) and Subramanya et al. (2010). These updates do not require the computation of derivatives, but if one makes changes to the different penalties in the graph objective (say, changing the KL-based terms in Equation 6.19 to JS divergences), they require modifications to the alternating minimization techniques, and are not straightforward to derive.

In the following two sections, we compare the six graph objective functions in Table 6.1 with the two baseline objectives. We first examine a part-of-speech lexicon expansion problem as a benchmark scenario, followed by extensive examination of a FrameNet predicate lexicon expansion problem. For the latter, we investigate whether semi-supervised lexicons can improve frame-semantic parsing.

6.3 Experiments: POS Lexicon Expansion

We expand a POS lexicon for word types with a context word on each side, using distributional similarity in an unlabeled corpus and few labeled trigrams. This lexicon contains trigrams with POS tags for the middle word of the trigram, each assigned a score indicating how strongly a POS tag associates with the middle word, given its left and right context. Experiments in this section do not relate to the semantic analysis problems of our interest, but it exhibits the efficacy of the generic graph-based SSL techniques we have presented this is chapter.

6.3.1 Data and task

We constructed a graph over *word trigram types* as vertices, using co-occurrence statistics. Following Das and Petrov (2011) and Subramanya et al. (2010), similarity between two trigram types was computed by measuring the cosine distance between their empirical sentential context (pointwise mutual information) statistics. We define a symmetric similarity function $\mathsf{sim}(u, v)$ over two vertices in the trigram graph, where u and v correspond to different trigrams. This function is computed using co-occurrence statistics of the nine feature concepts given in Table 6.2.

Each feature concept is akin to a random variable and its occurrence in the text corresponds to a particular instantiation of that random variable. For each trigram type $x_2\ x_3\ x_4$ in a sequence $x_1\ x_2\ x_3\ x_4\ x_5$, we count how many times that trigram type co-occurs with the different instantiations of each concept, and compute the point-wise mutual information (PMI) between the each trigram type and its possible feature instantiations.[4] The similarity between two trigram types is given by summing over the PMI

[4]Note that many combinations are impossible giving a PMI value of 0; e.g., when the trigram type and

Description	Feature
Trigram + Context	$x_1\ x_2\ x_3\ x_4\ x_5$
Trigram	$x_2\ x_3\ x_4$
Left Context	$x_1\ x_2$
Right Context	$x_4\ x_5$
Center Word	x_3
Trigram $-$ Center Word	$x_2\ x_4$
Left Word + Right Context	$x_2\ x_4\ x_5$
Left Context + Right Word	$x_1\ x_2\ x_4$
Suffix	HasSuffix(x_3)

Table 6.2: Various features used for computing edge weights between trigram types. A trigram $\mathbf{x}$ is denoted by $x_2\ x_3\ x_4$.

values over feature instantiations that they have in common. This is similar to stacking the different feature instantiations into long (sparse) vectors and computing the cosine similarity between them.

This similarity score resulted in the symmetric weight matrix $\mathbf{W}$, defining edge weights between pairs of graph vertices. $\mathbf{W}$ is thresholded so that only the K nearest neighbors for each vertex have similarity greater than zero, giving a sparse graph. We set $K = 8$ as it resulted in the sparsest graph which was fully connected.[5] For this task, Y is the set of 45 POS tags defined in the Penn Treebank (Marcus et al., 1993), and the measure q_i for vertex i (for trigram type $\mathbf{x}_i$) corresponds to the set of tags that the middle word of $\mathbf{x}_i$ can evoke. The trigram representation, as in earlier work, helps reduce the ambiguity of POS tags for the middle word, and helps in graph construction. The graph was constructed over all trigram types appearing in Sections 00–21 (union of the training and development sets used for POS tagging experiments in prior work) of the WSJ section of the Penn Treebank, but co-occurrence statistics for graph construction were gathered from a million sentences drawn from the English Gigaword corpus (Graff, 2003). This graph contained 690,705 vertices. Figure 6.2 shows an excerpt from this graph.

the feature instantiation don't have words in common.

[5]Our proposed methods can deal with graphs containing disconnected components perfectly well. Runtime is asymptotically linear in K for all objectives considered here.

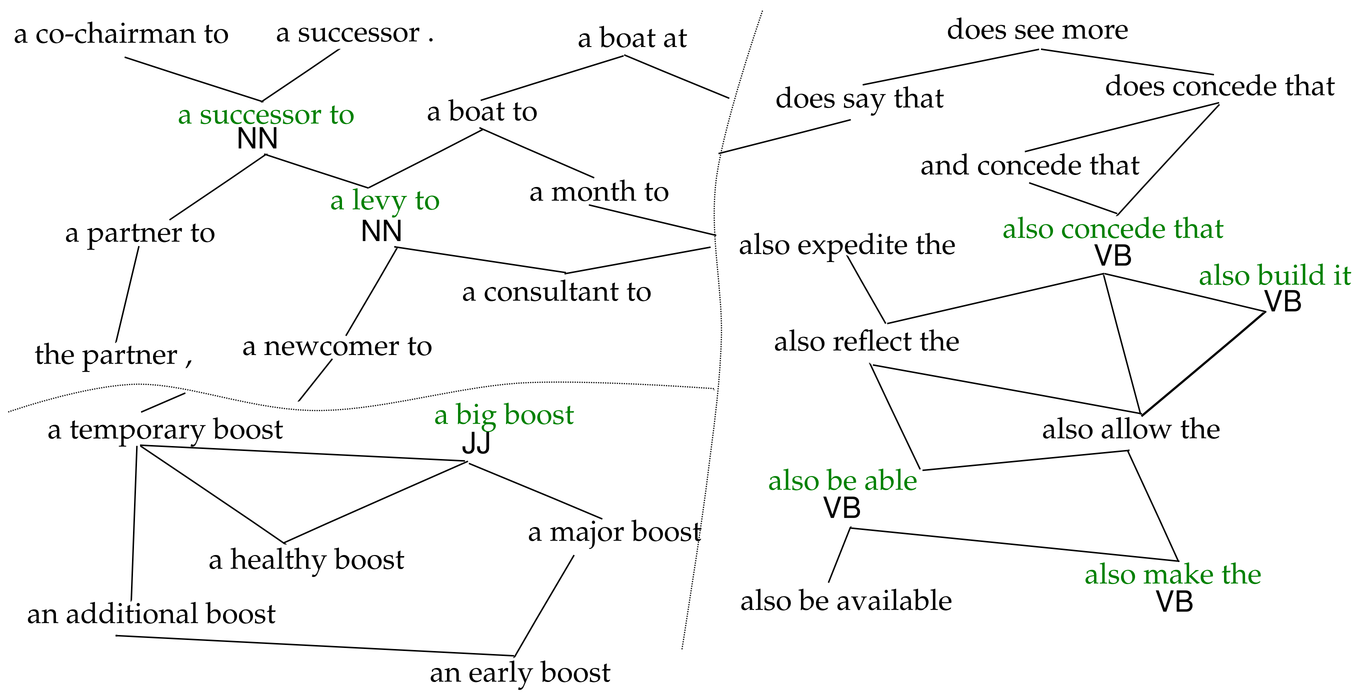


Figure 6.2: An excerpt from the graph constructed using trigram types taken from Sections 00-21 of the Wall Street Journal section of the Penn treebank. The top left portion of the figure shows trigrams whose middle word take the NN tag while the bottom left portion shows trigrams middle words of which take the JJ tag. The section on the right shows trigrams that take the VB tag of the Penn treebank. The green trigram types indicate trigrams extracted from labeled data (the transduction set containing 24,000 labeled vertices) while the black ones denote types on which we desire to induce tag distributions. Weights on the graph edges are not shown to simplify the figure.

Given a graph $\mathcal{G}$ with m vertices, we assume that the tag distributions $\hat{\boldsymbol{q}}$ for l labeled vertices are also provided. Our goal is to find the best set of measures $\boldsymbol{q}$ over the 45 tags for all vertices in the graph. Das and Petrov (2011) and Subramanya et al. (2010) used a similar lexicon for POS domain adaptation and POS induction for resource-poor languages; such applications of a POS lexicon are out of scope here; we consider only the lexicon expansion problem and do an intrinsic evaluation at a type-level to compare the different graph objectives.

6.3.2 Experimental details

To evaluate, we randomly chose 6,000 out of the 690,705 types for development. From the remaining types, we randomly chose 588,705 vertices for testing. These vertices were chosen after graph construction over the entire set of trigram types in our dataset. This left us with 96,000 types from which we created sets of different sizes containing 3,000, 6,000, 12,000, 24,000, 48,000 and 96,000 *labeled* types, creating 6 increasingly easy transduction settings. The development and the test types were kept constant for direct performance comparison across the six settings and our eight models.

After running inference for a hyperparameter setting, the measure q_i at vertex i was normalized to 1. Next, for all thresholds ranging from 0 to 1, with steps of 0.001, we measured the average POS tag precision and recall on the development data – this gave us the area under the precision-recall curve (prAUC), which is often used to measure performance on retrieval tasks.

Given a transduction setting and the final $\boldsymbol{q}*$ for an objective, hyperparameters μ and λ were tuned on the development set by performing a grid search, using prAUC. For the objectives using the uniform ℓ_2 and the maximum entropy penalties, namely **UGF**-ℓ_2, **UJSF**-ℓ_2, **NGF**-ℓ_2 and **NKLF**-ME, we chose λ from $\{0, 10^{-6}, 10^{-4}, 0.1\}$. For the rest of the models using sparsity inducing penalties, we chose λ from $\{10^{-6}, 10^{-4}, 0.1\}$.

This suggests that for the former type of objectives with uniform penalties, we allowed a zero unary penalty if that setting resulted in the best development performance, while for the latter type of models, we enforced a positive unary penalty. In other words, we examined whether the tuning procedure prefers $\lambda = 0$ for the non-sparse objectives, which would exhibit that uniform penalties hurt performance. In fact, $\lambda = 0$ was chosen in several cases for the graph objectives with uniform penalties indicating that uniformity hurts performance. We chose μ from $\{0.1, 0.5, 1.0\}$. We ran 100 rounds of iterative updates for all 8 graph objectives.

6.3.3 Type-level evaluation

To measure the quality of the lexicons, we perform type-level evaluation using area under the precision-recall curve (prAUC). The same measure (on development data)

$\lvert\mathcal{D}_l\rvert$:	3K	6K	12K	24K	48K	96K
NGF-ℓ_2	0.208	0.219	0.272	0.335	0.430	0.544
NKLF-ME	0.223	0.227	0.276	0.338	0.411	0.506
UGF-ℓ_2	0.223	0.257	0.314	**0.406**	**0.483**	**0.564**
UGF-ℓ_1	0.223	0.257	0.309	**0.406**	**0.483**	0.556
UGF-$\ell_{1,2}$	0.223	0.256	0.313	0.403	0.478	0.557
UJSF-ℓ_2	**0.271**	0.250	0.310	0.364	0.409	0.481
UJSF-ℓ_1	0.227	0.257	**0.317**	0.369	0.410	0.481
UJSF-$\ell_{1,2}$	0.227	**0.258**	0.309	0.369	0.409	0.479

Table 6.3: Area under the precision recall curve for the two baseline objectives and our methods for POS tag lexicon induction. This is a measure of how well the type lexicon (for some types unlabeled during training) is recovered by each method. The test set contains 588,705 types. Bold indicates best results for a particular column.

was used to tune the two hyperparameters. Table 6.3 shows the results measured on 588,705 test vertices (the same test set was used for all the transduction settings). The general pattern we observe is that our unnormalized approaches almost always perform better than the normalized baselines. (The exception is the 3,000 labeled example case, where most unnormalized models are on par with the better baseline.) In scenarios with fewer labeled types, pairwise entropic penalties perform better than Gaussian ones, and the pattern reverses as more labeled types come available. This trend is the same when we compare only the two baselines. In four out of the six transduction settings, one of the sparsity-inducing graph objectives achieves the best performance in terms of prAUC, which is encouraging given that they generally produce smaller models than the baselines.

Overall, though, using sparsity-inducing unary factors seems to have a weak negative effect on performance. Their practical advantage is apparent when we consider the size of the model. After the induction of the set of measures $\boldsymbol{q}$ for all transduction settings and all graph objectives, we noticed that our numerical optimizer (LBFGS-B) often fails to assign 0 mass to several components, rather assigning extremely low positive values. This problem can be attributed to several artifacts, including not running all optimization procedures until convergence, but choosing 100 as a fixed number of maximum iterations. Hence, we use a global threshold of 10^{-6}, and treat any real value below this threshold to be zero. Figure 6.3 shows the number of non-zero components in $\boldsymbol{q}$ (or, the lexicon size) for the graph objectives that achieve sparsity (baselines **NGF**-

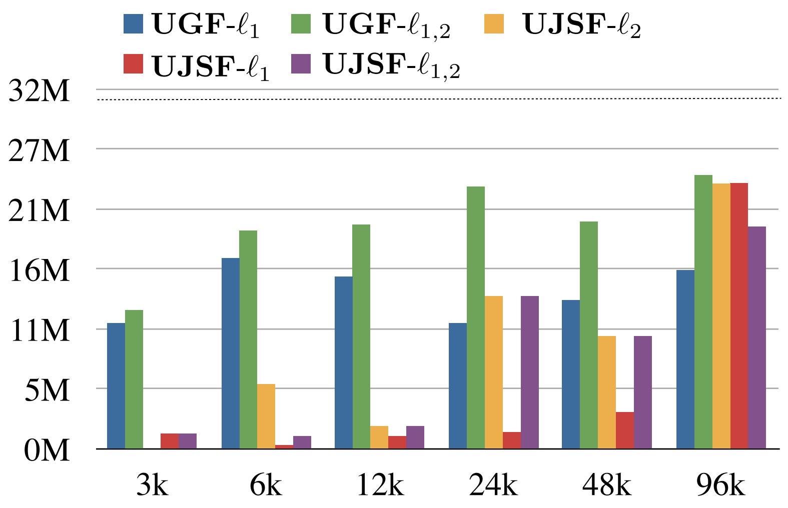


Figure 6.3: The number of non-zero components in $\boldsymbol{q}$ for five graph objective functions proposed in this work, plotted against various numbers of labeled datapoints. Note that **NGF**-ℓ_2, **NKLF**-ME and **UGF**-ℓ_2 produce non-zero components for virtually all $\boldsymbol{q}$, and are therefore not shown (the dotted line marks the maximally non-sparse solution, with 31,081,725 components). All of these five objectives result in sparsity. On average, the objectives employing entropic pairwise penalties with sparse unary penalties **UJSF**-ℓ_1 and **UJSF**-$\ell_{1,2}$ produce very sparse lexicons. Although **UGF**-ℓ_2 produces no sparsity at all, its entropic counterpart **UJSF**-ℓ_2 produces considerable sparsity, which we attribute to the quality of JS-divergence, which is used as a pairwise penalty.

ℓ_2 and **NKLF**-ME, plus our **UGF**-ℓ_2 are not expected to, and do not, achieve sparsity; surprisingly **UJSF**-ℓ_2 *does* and is shown). Even though the hyperparameters μ and λ in the graph objective functions were not tuned towards sparsity, we see that sparsity-inducing factors are able to achieve far more compact lexicons. Sparsity is desirable in settings where labeled development data for tuning thresholds is unavailable (e.g., Das and Petrov, 2011).

6.4 Experiments: Expansion of a Semantic Frame Lexicon

In a second set of experiments, we focus on expanding a lexicon that associates lexical units with semantic *frames* as labels. More concretely, each vertex in the graph corresponds to a lemmatized word type with its coarse part of speech, and the labels are frames from the FrameNet lexicon (Fillmore et al., 2003). We detail the lexicon expansion problem for this task in the following subsections. Specifically, we examine the problem of finding the possible set of frames for an unknown LU, and using that set for better frame identification for LUs unseen in supervised FrameNet data (see §5.3 for the details of our frame identification model).[6]

6.4.1 Graph Construction

We construct a graph with lexical units as vertices. For us, each LU corresponds to a lemmatized word or phrase appended with a coarse POS tag (identical to the representation in Chapter 5). We use two resources for graph construction. First, we take all the words and phrases present in a dependency-based thesaurus constructed using syntactic cooccurrence statistics (Lin, 1998).[7] To construct this resource, a corpus containing 64 million words was parsed with a fast dependency parser (Lin, 1993, 1994), and syntactic contexts were used to find similar lexical items for a given word or phrase. Lin separately treated nouns, verbs and adjectives/adverbs and the thesaurus contains three parts for each of these categories. For each item in the thesaurus, 200 nearest neighbors are listed with a symmetric similarity score between 0 and 1. We processed this thesaurus in two ways: first, we lowercased and lemmatized each word/phrase and merged entries which shared the same lemma; second, we separated the adjectives and adverbs into two lists from Lin's original list by scanning a POS-tagged version of the Gigaword corpus (Graff, 2003) and categorizing each item into an adjective or an adverb depending on which category the item associated with more often in the data. The second step was necessary because FrameNet treats adjectives and adverbs separately. At the end of this processing step, we were left with 61,702 units—approximately six times more than the LUs found in FrameNet annotations—each labeled with one of 4 coarse tags. We considered only the 20 most similar LUs for each LU, and noted Lin's similarity between two LUs v and u, which we call $\text{sim}_{DL}(v, u)$.

[6]It worth mentioning that we performed a preliminary set of experiments for improving frame identification by considering some extra features in our model (see §5.3.2) that look at semi-supervised word representations; in particular, we used Brown clusters (Brown et al., 1992) as additional features in our model without extensive tuning of cluster granularity. However, we did not see any improvements over our supervised model and resorted to the use of graph-based SSL techniques, which have shown promise in other natural language problems.

[7]Available at `http://webdocs.cs.ualberta.ca/~lindek/Downloads/sim.tgz`

The second component of graph construction comes from FrameNet itself. We scanned the exemplar sentences in FrameNet 1.5 and the training section of the full-text annotations that we use to train the probabilistic frame-semantic parser (see §5.1.2), and gathered a distribution over frames for each LU. For a pair of LUs v and u, we measured the Euclidean distance[8] between their frame distributions. This distance was next converted to a similarity score, namely, $\mathsf{sim}_{FN}(v, u)$ between 0 and 1 by subtracting each one from the maximum distance found in the whole data, followed by normalization. Like $\mathsf{sim}_{DL}(v, u)$, this score is symmetric. This resulted in 9,263 LUs, and again for each, we considered the 20 most similar LU. Finally, the overall similarity between two given LUs v and u was computed as:

$$\mathsf{sim}(v, u) = \alpha \cdot \mathsf{sim}_{FN}(v, u) + (1 - \alpha) \cdot \mathsf{sim}_{DL}(v, u)$$

Note that this score is symmetric because its two components are symmetric. The intuition behind taking a linear combination of the two types of similarity functions is as follows. We hope that distributionally similar LUs would have the same semantic frames because lexical units evoking the same set of frames appear in similar syntactic contexts. We would also like to involve the annotated data in graph construction so that it can eliminate some noise in the automatically constructed thesaurus. Let $\mathcal{K}(v)$ denote the K most similar LUs to LU v, under the score sim. Let the graph vertices corresponding to the LUs v and u be i and k respectively. Let the weight on the edge connecting i and k be w_{ik}, defined as:

$$w_{ik} = \begin{cases} \mathsf{sim}(v, u) & \text{if } v \in \mathcal{K}(u) \text{ or } u \in \mathcal{K}(v) \\ 0 & \text{otherwise} \end{cases} \tag{6.20}$$

The hyperparameter α was tuned by cross-validation and K was fixed to 10 (§6.4.3). The constructed graph contains 64,480 vertices, each corresponding to a LU, out of which 9,263 were drawn from the labeled data. The possible set of labels Y is the set of 877 frames defined in FrameNet. In this application, the measure q_i corresponds to the set of frames that the LU v (corresponding to vertex i) can evoke. The LUs drawn from FrameNet annotated data ($l = 9{,}263$) have frame distributions $\hat{q}_i$ with which the graph objectives are seeded. As mentioned earlier, we used the training section of the full text annotations part of FrameNet 1.5 to derive the seed LUs (see §5.1.2).

[8]This could have been replaced by an entropic divergence like KL- or JS-divergence, but we leave that exploration to future work.

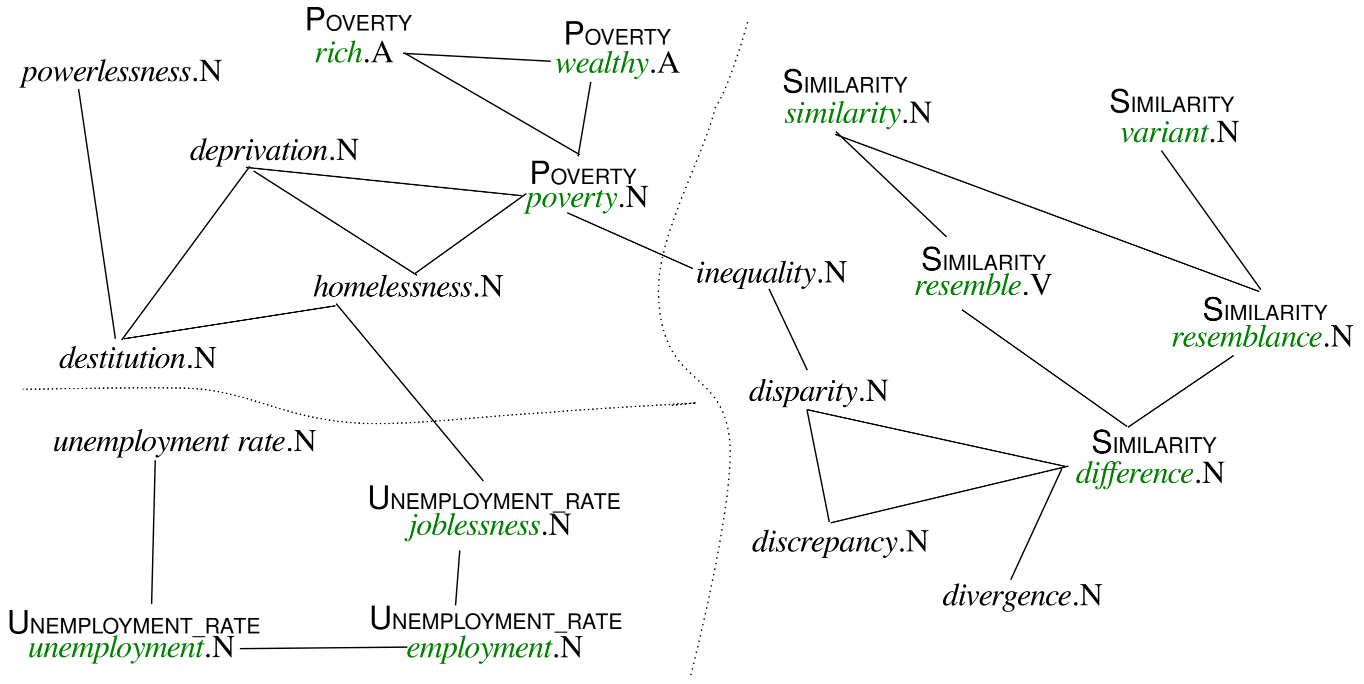


Figure 6.4: Excerpt from a graph over lexical units. Green LUs are observed in the FrameNet 1.5 data. Above/below them are shown the most frequently observed frame that they evoke. The black LUs are unobserved and graph propagation produces a distribution over most likely frames that they could evoke.

6.4.2 Constrained Frame Identification

Although (partial) frame identification accuracy on the FrameNet 1.5 dataset is 90.51% (Table 5.5), we found that the performance on portion of the test set containing only unseen LUs is only 46.62%. The latent variable frame identification model (see §5.3.2) that employs lexical-semantic features derived from WordNet strives to generalize performance on unknown LUs; however, this poor performance shows that a resource larger than WordNet is essential to improve coverage on unknown LUs. To this end, we use automatically induced frame distributions over unknown LUs using semi-supervised learning to constrain our frame identification model.

Equation 5.1 describes the inference rule for frame identification. We repeat the rule below:

$$f_i \leftarrow \operatorname*{argmax}_{f \in \mathcal{F}_i} \sum_{\ell \in \mathcal{L}_f} p_{\boldsymbol{\theta}}(f, \ell \mid t_i, \mathbf{x}) \tag{6.21}$$

Above, t_i is the i^{th} target in a sentence $\mathbf{x}$, and f_i is the corresponding evoked frame. In our constrained version, we use the above rule exactly, but with a minor twist. Recall from Section 5.3.2 that for targets with known lemmatized forms, $\mathcal{F}_i$ was defined to be the set of frames that associate with lemma t_i^l in the supervised data. For unknown lemmas, $\mathcal{F}_i$ was defined to be all the frames in the lexicon. If the LU corresponding to t_i is present in the graph, let it be the vertex $\underline{i}$. For such targets t_i covered by the graph, we redefine $\mathcal{F}_i$ as:

$$\mathcal{F}_i = \{f : f \in M\text{-best frames under } q^*_{\underline{i}}\} \tag{6.22}$$

Above, $q^*_{\underline{i}}$ is the final measure induced by a graph propagation algorithm for the LU corresponding to t_i. For targets t_i whose LUs are not present in the graph as well as in supervised data, we reinstate $\mathcal{F}_i$ to be the set of all frames. M is treated as a hyperparameter and is tuned using cross validation.

6.4.3 Experiments and Results

In this subsection, we describe the experiments conducted using the graph-based SSL techniques presented in the previous sections on the frame-semantic parsing task.

Experimental Setup

We measure frame disambiguation accuracy and frame-semantic parsing performance given gold targets, on the portion of the FrameNet 1.5 full text annotation test set that contains targets *unseen* in the FrameNet annotated data. We also present results on the whole test set. There are 144 unknown targets in the test data (see the details of the test data in §5.1.2).

Note that given the measure q_i over frames induced using graph-based SSL for vertex i, we truncate it to keep at most the top M frames that get the highest mass under q_i, only retaining those with non-zero values. If all components of q_i are zero, we remove vertex i from the lexicon, which is often the case in the sparsity-inducing graph objectives. A separate probabilistic model then disambiguates among the M filtered frames observing the sentential context of the LU's target instance (Equation 6.21). This can be thought of as combining type- and token-level information for inference. Argument identification follows in the same way as in §5.4 without any modification. We fixed K, the number of nearest neighbors for each vertex, to be 10. For each graph objective, μ, λ and M were chosen by five-fold cross-validation. Cross-validation was performed by dividing the supervised training set of FrameNet 1.5 into five folds. For each fold, accuracy on the unseen part of the fold's test portion was measured for hyperparameter tuning. We chose μ from $\{0.01, 0.1, 0.3, 0.5, 1.0\}$; λ was chosen from the same sets as the POS problem. The best α, the graph-construction parameter was tuned only for the **NGF**-ℓ_2 objective to limit the number of experiments. It was chosen as 0.2 and was fixed for rest of the experiments. M was automatically selected to be 2 for all graph objectives from $\{2, 3, 5, 10\}$.

Self-training Baseline

In our experiments, we consider a semi-supervised self-trained system for comparison with the graph-constrained models. For this system, we used the supervised frame identification system to label 70,000 sentences from the English Gigaword corpus with frame-semantic parses. For finding targets in a raw sentence, we used a relaxed target identification scheme, where we marked every target seen in the lexicon and all other words which were not prepositions, particles, proper nouns, foreign words and Wh-words as potential frame evoking units. This was done so as to find unseen targets and get automatic frame annotations on them. We appended these automatic annotations to the training data, resulting in 711,401 frame annotations, more than 36 times the annotated data. These data were next used to train a frame identification model.[9] This setup is very similar to Bejan (2009) who used self-training to improve frame identification.

[9]Note that we only self-train the frame identification model and not the argument identification model, which is fixed throughout. We ran self-training with smaller amounts of data, but found no significant difference with the results achieved with 711,401 frame annotations. As we observe in Table 6.4, in our case, self-training performs worse than the supervised model, and we do not hope to improve with even more data.

Results

Table 6.4 shows frame identification accuracy, both using exact match as well as partial match that assigns partial credit when a related frame is predicted. Performance on both unknown as well as all targets are shown. The final column presents lexicon size in terms of the set of truncated frame distributions (filtered according to the top M frames in q_i) for all the LUs in a graph. Firstly, note that for unknown targets the graph-based objectives outperform both the supervised model as well as the self-training baseline by a margin of $\sim 20\%$ accuracy points. The best model is **UJSF-**$\ell_{1,2}$, and its performance is significantly better than the supervised model ($p < 0.01$).

It also produces the smallest lexicon, using the sparsity inducing penalty. The improvements of the graph-based objectives are modest for the whole test set, but the best model still has statistically significant improvements over the supervised model ($p < 0.01$). For our objectives using pairwise Gaussian fields with sparse unary penalties, the accuracies are equal or better **NGF-**ℓ_2; however, the lexicon sizes are reduced by a few hundred to a few thousand entries. Massive reduction in lexicon sizes (as in the part-of-speech problem in §6.3) is not visible for these objectives because we throw out most of the components of the entire set of distributions $\boldsymbol{q}$ and keep only at most the top M (which is automatically chosen to be 2 for all objectives) frames per LU. Although a significant number of components in the whole distribution $\boldsymbol{q}$ are zeroed out, the M components for a LU tend to be non-zero for a majority of the lexical units.

Better results are observed for the graph objectives using entropic pairwise penalties in general; the objective **UJSF-**$\ell_{1,2}$ gives us the best absolute result by outperforming the baselines by strong margins, and also resulting in a tiny lexicon, less than half the size of the baseline lexicons. The tiny size can be attributed to the removal of LUs for which all frame components were zero ($q_i = \mathbf{0}$). Note that the **UJSF-**$\ell_{1,2}$ model outperforms **NGF-**ℓ_2 by a margin of 2.9% using partial frame matching, which is encouraging, but the difference is not quite significant ($p < 0.1$).[10]

The improvements of the graph-based models over the supervised system carry over to the full frame-semantic parsing performance (given gold targets). We show those results in Table 6.5. The trends are very similar to the frame identification results, and the best graph-based model **UJSF-**ℓ_2 produces statistically significant results over the supervised baseline for precision, recall and F_1 score for both unknown targets and the whole test set ($p < 0.001$).

[10]We also ran an experiment where we trained a frame identification model without the latent variable (see §5.3.4 for details) and used the frame distributions over unknown predicates from the **UJSF-**$\ell_{1,2}$ objective to constrain frame identification inference. This model gave us an exact match accuracy of 76.55% and a partial match accuracy of 86.78%, which are better than the vanilla supervised no latent-variable model (compare with the accuracy figures in §5.3.4). However, these accuracies fall short of the numbers reported in Table 6.4, exhibiting the fact that the latent-variable frame identification model along with the graph constraints perform the best on this task.

	UNKNOWN TARGETS		ALL TARGETS		
	exact frame matching	**partial frame matching**	**exact frame matching**	**partial frame matching**	Lexicon Size
Supervised	23.08	46.62	82.97	90.51	-
Self-training	18.88	42.67	82.27	90.02	-
NGF-ℓ_2	39.86	62.35	83.51	91.02	128,960
NKLF-ME	36.36	60.07	83.40	90.95	128,960
UGF-ℓ_2	37.76	60.81	83.44	90.97	128,960
UGF-ℓ_1	39.86	62.85	83.51	91.04	122,799
UGF-$\ell_{1,2}$	39.86	62.85	83.51	91.04	128,732
UJSF-ℓ_2	40.56	62.81	83.53	91.04	128,232
UJSF-ℓ_1	39.16	62.43	83.49	91.02	128,771
UJSF-$\ell_{1,2}$	**42.67**	**65.29**	**83.60**	**91.12**	**45,544**

Table 6.4: Exact and partial frame identification accuracy with the size of lexicon (in terms of non-zero frame components) used for frame identification. The portion of the test set with unknown targets contains 144 targets unseen in supervised data. Bold indicates best results. The **UJSF**-$\ell_{1,2}$ model produces statistically significant results ($p < 0.001$) for all metrics with respect to the supervised baseline for both the unseen targets as well as the whole test set. However, it is only weakly significant ($p < 0.1$) with respect to the **NGF**-ℓ_2 model for the unseen portion of the test set, when partial frame matching is used for evaluation. For rest of the settings, these two models are statistically indistinguishable. However, it is noteworthy that the **UJSF**-$\ell_{1,2}$ objective produces a much smaller lexicon in comparison to all the other graph objectives.

	Unknown Targets						All Targets					
	exact frame matching			**partial frame matching**			**exact frame matching**			**partial frame matching**		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Supervised	18.99	14.90	16.70	33.21	26.05	29.20	67.74	60.60	63.97	72.39	64.76	68.37
Self-training	14.60	11.41	12.81	29.09	22.73	25.52	67.07	60.01	63.43	71.83	64.27	67.84
NGF-ℓ_2	34.76	27.10	30.45	48.75	38.01	42.71	68.15	60.95	64.35	72.80	65.11	68.74
NKLF-ME	32.72	25.52	28.67	47.26	36.85	41.41	68.10	60.91	64.30	72.77	65.08	68.71
UGF-ℓ_2	33.20	26.15	29.26	47.63	37.51	41.97	68.10	60.93	64.32	72.77	65.10	68.72
UGF-ℓ_1	34.28	26.78	30.07	48.54	37.92	42.58	68.14	60.94	64.34	72.98	65.11	68.74
UGF-$\ell_{1,2}$	34.28	26.78	30.07	48.54	37.92	42.58	68.14	60.94	64.34	72.98	65.11	68.74
UJSF-ℓ_2	36.72	28.05	31.81	50.69	38.72	43.91	68.22	60.98	64.40	72.87	65.13	68.78
UJSF-ℓ_1	34.01	26.47	29.77	48.32	37.60	42.29	68.14	60.94	64.33	72.79	65.10	68.73
UJSF-$\ell_{1,2}$	**39.15**	**30.59**	**34.34**	**53.29**	**41.64**	**46.75**	**68.26**	**61.06**	**64.46**	**72.92**	**65.22**	**68.86**

Table 6.5: Frame-semantic parsing results using different graph-based SSL objectives. Note that we use gold targets for these experiments and evaluate frame identification and argument identification together. Bold indicates best results. In terms of precision, recall and F_1 score, the **UJSF**-$\ell_{1,2}$ model is statistically significant compared to the supervised baseline using both exact and partial frame matching, for both the unseen targets and the full test set ($p < 0.001$). For both the unseen targets and the whole test set, in terms of precision and F_1 score measured with partial frame matching, the **UJSF**-$\ell_{1,2}$ model is statistically significant over the **NGF**-ℓ_2 model ($p < 0.05$). For recall with partial frame matching, and for all the three metrics with exact frame matching, these two graph objectives are statistically indistinguishable, although the **UJSF**-$\ell_{1,2}$ produces a much smaller lexicon.

LU= *discrepancy*.N	LU= *contribution*.N	LU= *print*.V	LU= *mislead*.V
∗Similarity	∗Giving	∗Text_creation	Experiencer_obj
Natural_features	Money	Sending	∗Prevarication
Prevarication	Commitment	Dispersal	Manipulate_into_doing
Quarreling	Assistance	Reading	Compliance
Duplication	Earnings_and_losses	Statement	Evidence

LU= *abused*.A	LU= *maker*.N	LU= *inspire*.V	LU= *failed*.A
Offenses	Commerce_scenario	Cause_to_start	Success_or_failure
Killing	∗Manufacturing	Experiencer_obj	∗Successful_action
Compliance	Businesses	∗Subjective_influence	Unattributed_information
Differentiation	Behind_the_scenes	Evoking	Piracy
Committing_crime	Supply	Attempt_suasion	Want_suspect

Table 6.6: Top 5 frames according to the graph posterior distributions $\boldsymbol{q}^*$ with the graph objective **NGF**-ℓ_2 for eight LUs: *discrepancy*.N, *contribution*.N, *print*.V, *mislead*.V, *abused*.A, *maker*.N, *inspire*.V and *failed*.A. None of these LUs were present in the supervised FrameNet data. $*$ marks the correct frame, according to the target instances in test data (each of these LUs appear only once in test data as targets). For the LU *abused*.A, the correct frame is not present in the top 5 frames under the distribution. Moreover, for the LU *inspire*.V, the correct frame is listed at position 3, which is unavailable to the frame identifier due to the selected hyperparameter $M = 2$.

LU= *discrepancy*.N	LU= *contribution*.N	LU= *print*.V	LU= *mislead*.V
∗Similarity	∗Giving	∗Text_creation	∗Prevarication
Non-commutative_statement	Commerce_pay	State_of_entity	Experiencer_obj
Natural_features	Commitment	Dispersal	Manipulate_into_doing
	Assistance	Contacting	Reassuring
	Earnings_and_losses	Reading	Evidence

LU= *abused*.A	LU= *maker*.N	LU= *inspire*.V	LU= *failed*.A
	∗Manufacturing	Cause_to_start	∗Successful_action
	Businesses	∗Subjective_influence	Successfully_communicate_message
	Commerce_scenario	Objective_influence	
	Supply	Experiencer_obj	
	Being_active	Setting_fire	

Table 6.7: Top 5 frames (if there are $\geq$ 5 frames with mass greater than zero) according to the final graph posterior measures $\boldsymbol{q}^*$ with the graph objective **UJSF**-$\ell_{1,2}$ for the same LUs as Table 6.6. Note that for the first LU *discrepancy*.N only three frames get non-zero mass after optimizing the graph objective. Similarly, for the LU *failed*.A, only two frames get non-zero mass. For the LU *abused*.A, all frames get zeroed out because the model is not confident about any frames resulting in the removal of this LU from the lexicon; this is in contrast to the result in Table 6.6, where the correct frame is not present in the top five frames shown. Finally, it is noteworthy that this objective ranks the correct frame higher in the frame list for most LUs in comparison to the **NGF**-ℓ_2 objective.

In Tables 6.6 and 6.7 we show the top frames under the final induced set of measures $\boldsymbol{q}^*$ for eight unseen LUs for the **NGF-**ℓ_2 and the **UJSF-**$\ell_{1,2}$ graph objectives respectively. The latter objective often ranks the correct frames higher than the former. For the *discrepancy*.N LU, note that the latter objective assigns non-zero mass to only three frames. For several such LUs, this objective function assigns non-zero mass to a few frames only, thus leading to sparsity. In Figure 6.5, we plot the frame identification accuracy for the **NGF-**ℓ_2 objective function against the hyperparameter M, which is the number of top frames that are scored by the frame identification model. We notice that the accuracy peaks at $M = 2$ and then falls off; this is the value that is automatically selected with cross-validation for all objective functions. Figure 6.6 shows an example where the graph-based model **UJSF-**$\ell_{1,2}$ corrects an error made by the supervised model for the unseen LU *discrepancy*.N, both for frame identification and full frame-semantic parsing.

Note about Self-Training

For the frame identification task, Bejan (2009) found self-training to improve overall frame identification performance. Why does self-training *not* work in our setup? We conjecture that this may happen because of several reasons. First, Bejan (2009) evaluated using five-fold cross-validation, used an smaller dataset, and reported a micro-averaged accuracy measure. Hence, there is a mismatch between our experimental setup and his. However, let us compare absolute frame identification accuracy values despite the differences in datasets used for model training. The best result reported in Bejan (2009) is 84.73%, which corresponds to self-training. This is an improvement over a supervised model, which is 76.1% accurate. Our purely supervised model achieves an accuracy of 90.51%, while the graph-based systems do even better. Hence, our baseline supervised model is very powerful in comparison to the supervised counterpart of Bejan; therefore it is unclear whether self-training can improve over that powerful baseline. Second, our self-training model is different from Bejan's in one fundamental aspect. Our goal is to improve the coverage of our frame-semantic parser. Hence, we used a relaxed strategy to identify targets in unlabeled data, and then used the original supervised labels along with the automatic labels to self-train another model. However, Bejan only considers polysemous targets from annotated data, which he finds in unlabeled data, labels them with his supervised model, and uses them for self-training. Thus, we cannot directly compare our self-training setup to his. Third, finally, our self-trained model does poorly on known targets, which the graph-based models never do, because they do not alter the prediction of the supervised system for such targets. We conjecture that this happens because its mistakes on even known targets during self-training introduce further noise, bringing about a kind of "semantic drift."

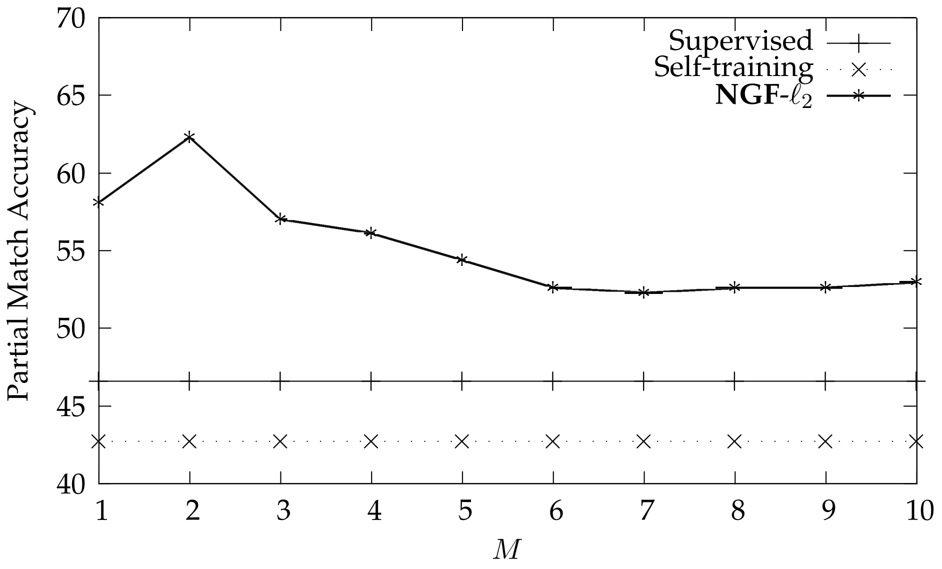


Figure 6.5: This graph shows the variation of frame identification accuracy using partial matching of frames for the constrained model using the **NGF-ℓ_2** graph objective, with M, the top frames that are most heavily weighted under the set of frame distributions induced by the SSL algorithm. This is measured on our test set. Note, that there is a rise in accuracy for $M = 2$, which is the value chosen by cross-validation for all graph objectives. This also conforms with the average frame ambiguity of 1.20 for a LU type in supervised data.

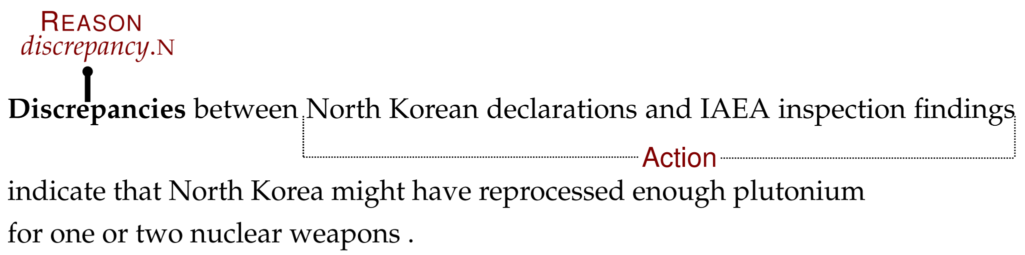
6.5 Discussion

In this chapter, we have made novel contributions from two perspectives. First, on the machine learning side, we presented models that contribute the following:

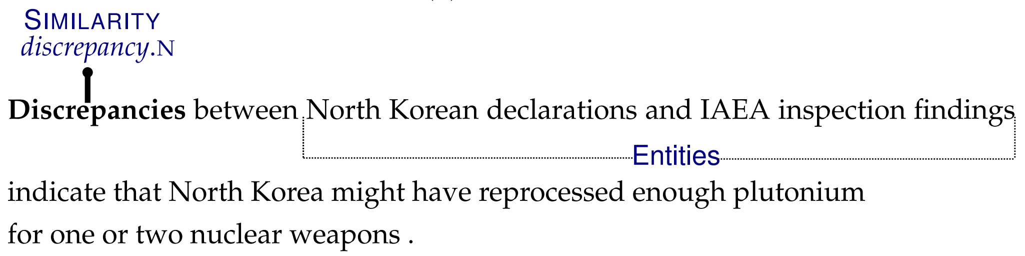
1. A family of graph-based SSL objective functions that incorporate penalties encouraging sparse measures at each graph vertex, leading to noisily constructed lexicons of small size.

2. Our methods relax the oft-used assumption that the measures at each vertex form a normalized probability distribution, making optimization and the use of complex penalties easier than in prior work.

3. Optimization is also easy when there are additional terms in a graph objective


REASON
discrepancy.N
Discrepancies between North Korean declarations and IAEA inspection findings
Action
indicate that North Korea might have reprocessed enough plutonium
for one or two nuclear weapons .

(a)


SIMILARITY
discrepancy.N
Discrepancies between North Korean declarations and IAEA inspection findings
Entities
indicate that North Korea might have reprocessed enough plutonium
for one or two nuclear weapons .

(b)

Figure 6.6: (a) Output of the supervised frame-semantic parsing model (Chapter 5) on the target corresponding to the LU *discrepancy*.N. The output is incorrect. (b) Output using the constrained frame identification model that takes into account the graph-based frame distributions over unknown targets. In this particular example, the **UJSF**-$\ell_{1,2}$ graph objective is used. This output matches the gold annotation. The LU *discrepancy*.N is unseen in supervised FrameNet data.

suited to a specific problem; our generic optimizer would simply require the computation of new partial derivatives, unlike prior work that required specialized techniques for a novel objective function, for example using custom update rules in an alternating minimization procedure (see Subramanya and Bilmes (2009)).

Second, we presented experiments on two natural language lexicon learning problems, which show that our methods produce comparable and often better performance with respect to state-of-the-art graph-based SSL methods, and also result in much smaller lexicons, which are easy to store and use in downstream applications. In particular, we have observed the following:

1. We presented experiments on a benchmark lexicon learning problem for POS tagging, where we observed that our novel objectives result in comparable or better

performance in terms of the prAUC metric with respect to established graph-based SSL objectives (Zhu et al., 2003; Subramanya and Bilmes, 2009).

2. On a more realistic task of frame-semantic parsing, a problem in which we are particularly interested in this dissertation, we show that not only does standard graph-based SSL techniques result in improvements over a supervised model on out-of-domain targets, our novel graph-based SSL objectives result in even better performance, also resulting in smaller lexicons.

The improved frame-semantic parser with graph-based constraints that help generalize the parser to unseen and out-of-domain targets is part of the SEMAFOR 2.1 package, available at `http://www.ark.cs.cmu.edu/SEMAFOR`.

Chapter 7

Conclusion and Future Work

In this chapter, we summarize the contributions of this thesis, and concentrate on the various possible future directions, given the research material presented in the previous chapters.

7.1 Novel Contributions of the Thesis

In this dissertation, we have looked at two semantic analysis problems: paraphrase identification and frame-semantic parsing. A major focus has been on the latter, it being a natural language understanding task which has the potential of having considerable impact on NLP applications; this research has resulted in a tool that the human language technologies community is using both for further study and in applications. We summarize our contributions:

1. We use a monolingual translation model, called a *quasi-synchronous grammar* to model the problem of paraphrase identification. The model employs loose syntactic and lexical-semantic transformations from one sentence to another, to recognize a paraphrase. Our work resulted in state-of-the-art results on a benchmark dataset. The model uses latent variables to capture alignments between words of a potential sentence pair; analysis of these latent variables at inference time gives interpretable explanation of the transformation process. Improvements over our model have led to even better results on this benchmark (Chang et al., 2010; Socher et al., 2011).

2. We present supervised models for the problem of shallow semantic parsing using the theory of frame semantics. We present two statistical models, trained probabilistically, that model the semantic frame disambiguation problem and the argument identification problem. Compared to prior work, we use fewer classifiers to

solve the problem, and also use latent variables to increase coverage of the parser. In comparison to the general semantic role labeling literature, our argument identification relies on only one model, while the norm has been the use of two models for filtering potential arguments, and then labeling them. Our results are the state of the art, evaluated on the SemEval 2007 benchmark dataset; we have also presented results on newer data released as part of the FrameNet 1.5 lexicon.

3. We augment our supervised models with a collective argument identification strategy that relies on an exact dual decomposition algorithm. This algorithm respects various linguistic constraints on the output of the frame-semantic parser, such as relationships between pairs of semantic roles. Our algorithm is able to respect these constraints during parsing, and maintains parsing speeds that are significantly faster than proprietary integer linear program solvers.

4. Finally, we present semi-supervised lexicon expansion techniques that are extensions of popular graph-based semi-supervised learning algorithms. When applied to frame-semantic parsing, these techniques are able to expand the predicate lexicon of FrameNet to lexical items absent in supervised data. This results in significantly better frame-semantic parsing performance on unseen, out-of-domain lexical units. Moreover, our novel graph-based semi-supervised learning algorithms produce lexicons that are much smaller in size in comparison to state-of-the-art techniques, but result in comparable, and often better results in NLP tasks.

7.2 Future Work

Here, we concentrate on a few promising future directions of research that can be extensions of the work presented in this dissertation. First, we focus on a possible modeling improvement for paraphrase identification, and then consider future avenues of research that relate to the frame-semantic parsing models presented in the prior chapters.

7.2.1 Frame-Semantic Representations for Paraphrase Identification

In our paraphrase model, we leverage syntactic parses in the form of dependency trees, latent alignments between the nodes of the trees of a sentence pair, and lexical-semantic relationships between words that are aligned; these lexical-semantic relationships are drawn from WordNet, which has significant coverage for English, the language at hand. However, we believe that semantic structures beyond just lexical-semantic relationships

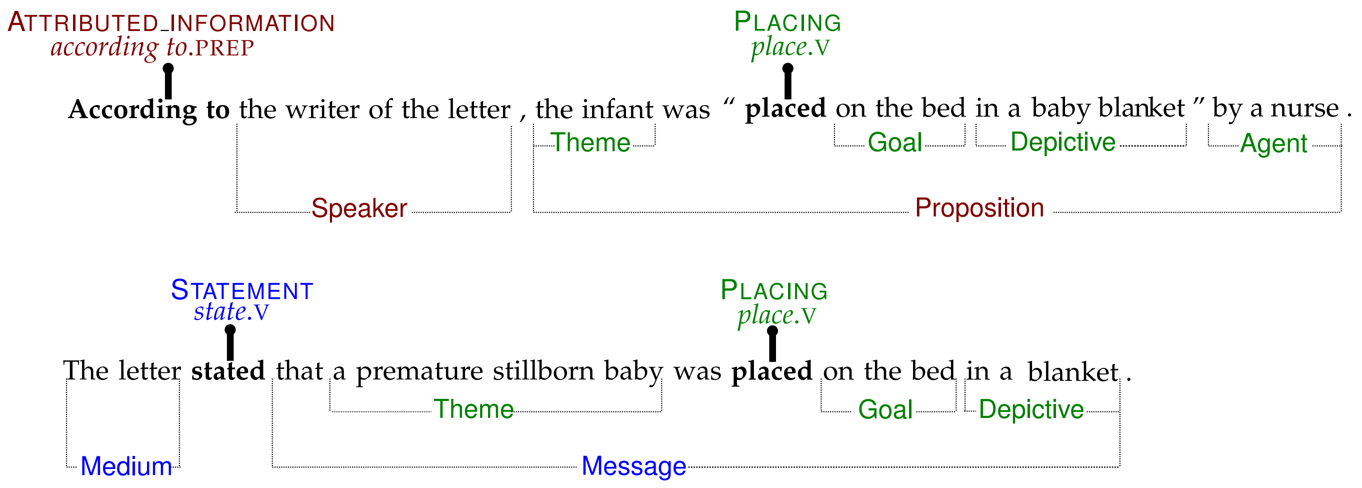

ATTRIBUTED_INFORMATION
according to.PREP
PLACING
place.V
According to the writer of the letter , the infant was " placed on the bed in a baby blanket " by a nurse .
Theme
Goal
Depictive
Agent
Speaker
Proposition
STATEMENT
state.V
PLACING
place.V
The letter stated that a premature stillborn baby was placed on the bed in a blanket .
Theme
Goal
Depictive
Medium
Message

Figure 7.1: A true paraphrase pair from the Microsoft Research Paraphrase Corpus (Dolan and Brockett, 2005). We also show partial, manually annotated frame-semantic parses on each sentence. Note that the PLACING frame is evoked in both sentences with the core roles Theme and Goal being instantiated in both.

can benefit the paraphrase identification task. Prior work has attempted to use logical form structures for textual entailment (Bos and Markert, 2005) with limited success.

Qiu et al. (2006) used PropBank-style semantic role labeling to improve paraphrase identification, in a shallow feature-based classifier. Although our presented paraphrase model outperforms this work in terms of accuracy, it exhibits the benefits of features derived from shallow semantic parses. We conjecture that frame-semantic analysis of individual sentences in a candidate paraphrase pair could provide valuable signals for true paraphrase recognition. In Figure 7.1, we show a true paraphrase sentence pair from the Microsoft Research Paraphrase Corpus. We also show partial frame-semantic parses of both sentences. Note that the frame PLACING is evoked in both sentences by the targets corresponding to the LU *place*.V. Moreover, the core roles Theme and Goal also appear in both sentences. While the arguments for Goal are identical in both sentences, the head words of the spans filling up the Theme role are synonyms of each other. The non-core role Depictive also appears in both sentences, with spans with identical head words. Other than this closely aligning predicate-argument structure, the frame ATTRIBUTED_INFORMATION is evoked in the first sentence and the STATEMENT is evoked in the second. This also is indicative of similar meaning across sentences because the frame ATTRIBUTED_INFORMATION *uses* the STATEMENT frame, according to the creators of the FrameNet. The *uses* relationship, which is a valid inter-frame relationship defined within FrameNet, demonstrates that a frame refers to the meaning of a parent frame.

It is straightforward to incorporate frame-semantic structures into the paraphrase model we presented in Chapter 4. One way would be to use the frame and role labels as features at each head word of a subtree of the sentence pair's dependency parses and use our current model with these additional features at alignment sites. Another way would be to use both syntactic and frame-semantic information on each sentence to create a graph structure, and model quasi-synchronous transformation between the two.

7.2.2 Training Frame-Semantic Parsers for Multiple Languages

As part of this dissertation, we have released the software necessary to train and test the presented frame-semantic parsing models. However, our experiments have focused only on English, as benchmarks and strong baselines exist for English. However, a straightforward extension of the current work would be training and testing models in various languages for which the corresponding FrameNet lexicon and full-text annotations exist. There exists such corpora for several languages. German is an example, for which the SALSA project has annotated a large frame-based lexicon and has frame-semantic annotations on free text (Burchardt et al., 2006). The annotations which are being carried out as part of the SALSA project are being done on the TIGER depen-

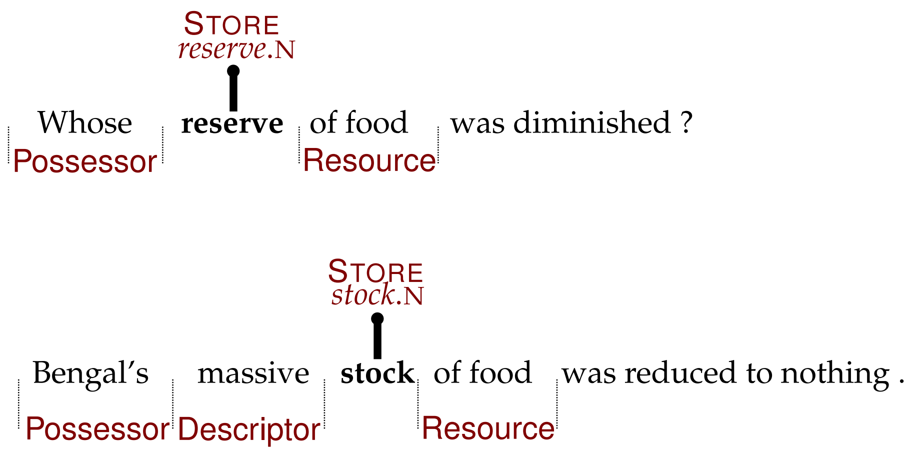


Figure 7.2: A factoid question and a sample answer with corresponding manually annotated frame-semantic parses. The pair has nearly isomorphic parses, with the answer having one extra non-core argument that fills up the Description role.

dency treebank (Brants et al., 2002), which provides gold syntactic trees; this has the potential of resulting in better quality frame-semantic parses than in English.

Brazilian Portuguese also boasts a FrameNet-style lexical database (Salomão, 2009) that has been developed in collaboration with the FrameNet team at ICSI, University of California, Berkeley. Although limited in size, this dataset is an ongoing effort in creating a language-specific FrameNet-style lexicon. A similar lexicon, much larger in size, also exists for Spanish (Subirats, 2004). A dataset containing several thousand annotated sentences is also available as part of this Spanish dataset, thus enabling the possibility of developing a Spanish frame-semantic parser. Finally, a moderately-sized lexicon for Japanese is also available (Ohara, 2011) with full-text annotations which could be used for the development of automatic frame-semantic parser for Japanese.

7.2.3 Multilinguality and Frame-Semantic Parsing

Although the universality of semantic frames are roles are controversial, there has been effort in projecting the information present in FrameNet, including the knowledge of lexical units into other languages. Padó and Lapata (2005a) present a model for the projection of the English FrameNet lexicon into German data and evaluate the projection method against gold standard annotations present in the SALSA corpus (Burchardt et al., 2006). More recently, Annesi and Basili (2010) present a similar method and eval-

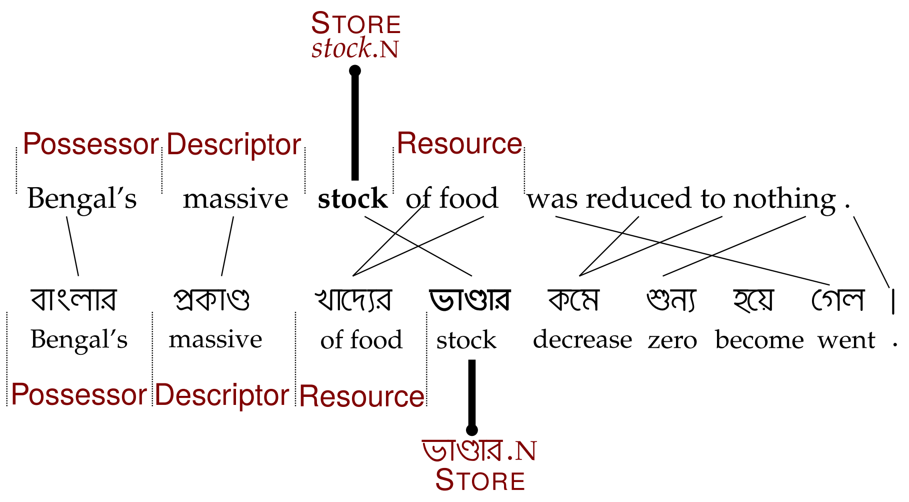

STORE
stock.N
Possessor
Descriptor
Resource
Bengal's massive stock of food was reduced to nothing .
বাংলার প্রকাণ্ড খাদ্যের ভাণ্ডার কমে শুন্য হয়ে গেল ।
Bengal's massive of food stock decrease zero become went .
Possessor
Descriptor
Resource
ভাণ্ডার.N
STORE

Figure 7.3: (Manually) aligned sentence pair in English and Bengali with a partial frame-semantic parse. Note that the exact frame-semantic parse for the lexical unit *stock*.N is reflected in the Bengali translation. The glosses for the Bengali sentence is shown below every Bengali word.

uate projection performance in Italian. A wider variety of languages can be considered and via parallel data, and automatic lexicons that resemble English FrameNet can be constructed. Following Das and Petrov (2011), who demonstrated a technique of lexicon projection from English to several other languages for POS tagging, one might construct semantic lexicons in other languages.

Machine translation is another multilingual NLP application that might benefit from frame-semantic parsing. Figure 7.3 shows a Bengali translation of an English sentence that we have considered before. Partial frame-semantic parses on both sides of the sentence pair show that frame-semantic structures are preserved. Alignments show that the arguments filling up identical semantic roles have direct correspondence. Frame-semantic parses can be used as features in a translation model. It is impractical to assume that frame-semantic parses are possible to extract for any language – however, for language pairs with English on one side could benefit from English frame-semantic features.

Finally, multilingual information extraction is possible only through English frame-semantic parses and translations of English sentences into multiple languages. This is especially possible for related news stories and other multilingual web data pertaining to related information. Via word alignments, semantic arguments in non-English sentences can be extracted through frame-semantic parse projection from English.

7.2.4 Frame-Semantic Parsing for NLP Applications

In our introduction (§1) we discussed briefly several possible applications of frame-semantic parsing in language technologies. Question answering is one such application into which frame-semantic parses can be incorporated as part of the preprocessing steps normally undertaken. Frame-semantic information can be useful either as features or constraints. Bilotti et al. (2007) used PropBank-style semantic role information to enable structured information retrieval for question answering. A straightforward extension of that work could as well incorporate information that looks at semantic frames and roles retrieved from frame-semantic parses, that label more types of syntactic categories as predicates than only verbs or nouns, and also use explicit role labels unlike PropBank semantic roles.

Figure 7.2 shows a factoid question and a candidate sentence that contains the true answer. Note that despite lexical variation at the surface, the two lexical units *reserve*.N and *stock*.N have the same frame STORE. The Wh-word in the question fulfills the Possessor role, which may indicate that one must look for similar roles in the passages containing the answer. Indeed, the phrase "Bengal 's" fulfills the Possessor role in the candidate sentence, and gives us the answer. Although in this question-answer pair, the predicate-argument structures are nearly isomorphic, structural correspondences may be loose, and it is possible to treat frame-semantic information as features, instead of treating it as a source of hard constraints.

Frame-semantic parsing can might be used for information extraction. Here are some example sentences containing targets that evoke the STORE frame and instantiate several semantic roles, stating facts about the world:

1. $\underline{\text{Bengal's}}_{\text{Possessor}}$ $\underline{\text{massive}}_{\text{Descriptor}}$ $\underset{\textit{stock}\text{.N}}{\overset{\text{STORE}}{\textbf{stock}}}$ $\underline{\text{of food}}_{\text{Resource}}$ was reduced to nothing.
2. In 1997, $\underline{\text{France's}}_{\text{Possessor}}$ $\underset{\textit{stock}\text{.N}}{\overset{\text{STORE}}{\textbf{stock}}}$ $\underline{\text{of unirradiated civil plutonium}}_{\text{Resource}}$ increased to 72 tons.
3. In 2004, there was also ample discussion concerning $\underline{\text{South Africa's}}_{\text{Possessor}}$ $\underline{\text{dwindling}}_{\text{Descriptor}}$ $\underline{\text{coal}}_{\text{Resource}}$ $\underset{\textit{reserve}\text{.N}}{\overset{\text{STORE}}{\textbf{reserves}}}$ and its need for additional nuclear power generation.

These predicate-argument structures can be used for information extraction, treating each frame as a relational database (say, STORE) with each role corresponding to a column in the database. Hence, from the example sentences above, the Possessor column of the database would contain the entities Bengal, France and South Africa. Confidence estimates from the frame-semantic parser can be used to create weighted entries in the database. Open information extraction systems, for example the "Never Ending Language Learner" (Carlson et al., 2010) can also benefit from a frame-semantic parser by using the n-ary relationships between phrases that fulfill different roles of annotated frame-semantic structures, to bootstrap and find similar sets of phrases.

Overall, the presented work shows promise from the perspective of future research – the models presented in this work are amenable to extensions that might result in better performance on the tasks' benchmarks, and also might be useful in several other natural language processing applications. The presented techniques, especially the modeling of latent structure and the generic semi-supervised learning methods might be used in other interesting problems in computational semantics, and other natural language understanding problems to significant benefit.

Bibliography

Andrew, G. and Gao, J. (2007). Scalable training of L1-regularized log-linear models. In *Proceedings of ICML*. [104]

Annesi, P. and Basili, R. (2010). Cross-lingual alignment of FrameNet annotations through hidden Markov models. In *Proceedings of CICLing*. [129]

Auli, M. and Lopez, A. (2011). A comparison of loopy belief propagation and dual decomposition for integrated CCG supertagging and parsing. In *Proceedings of ACL*. [75]

Baker, C., Ellsworth, M., and Erk, K. (2007). SemEval-2007 Task 19: Frame semantic structure extraction. In *Proceedings of SemEval*. [20, 50, 56]

Baluja, S., Seth, R., Sivakumar, D., Jing, Y., Yagnik, J., Kumar, S., Ravichandran, D., and Aly, M. (2008). Video suggestion and discovery for Youtube: taking random walks through the view graph. In *Proceedings of WWW*. [95]

Bannard, C. and Callison-Burch, C. (2005). Paraphrasing with bilingual parallel corpora. In *Proceedings of ACL*. [13]

Bar-Haim, R., Dagan, I., Dolan, B., Ferro, L., Giampiccolo, D., Magnini, B., and Szpektor, I. (2006). The second PASCAL recognising textual entailment challenge. In *Proceedings of the Second PASCAL Recognising Textual Entailment Challenge*. [1, 11]

Barzilay, R. and Lee, L. (2003). Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. In *Proceedings of NAACL*. [13, 34]

Barzilay, R. and McKeown, K. R. (2001). Extracting paraphrases from a parallel corpus. In *Proceedings of ACL*. [13]

Bejan, C. A. (2009). *Learning Event Structures From Text*. PhD thesis, The University of

Texas at Dallas. [115, 121]

Bengio, Y., Delalleau, O., and Le Roux, N. (2006). Label propagation and quadratic criterion. In Chapelle, O., Schölkopf, B., and Zien, A., editors, *Semi-Supervised Learning*, pages 193–216. MIT Press. [94]

Berant, J., Dagan, I., and Goldberger, J. (2011). Global learning of typed entailment rules. In *Proceedings of ACL*. [13]

Berger, A. (1996). A brief maxent tutorial. [27]

Bikel, D. M., Schwartz, R., and Weischedel, R. M. (1999). An algorithm that learns what's in a name. *Machine Learning*, 34(1). [39]

Bilotti, M. W., Ogilvie, P., Callan, J., and Nyberg, E. (2007). Structured retrieval for question answering. In *Proceedings of SIGIR*. [7, 24, 131]

Boas, H. C. (2002). Bilingual FrameNet dictionaries for machine translation. In *Proceedings of LREC*. [21]

Bos, J. and Markert, K. (2005). Recognising textual entailment with logical inference. In *Proceedings of HLT*. [1, 7, 12, 128]

Bottou, L. (2003). Stochastic learning. In *Advanced Lectures on Machine Learning*. [28, 69]

Brants, S., Dipper, S., Hansen, S., Lezius, W., and Smith, G. (2002). The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*. [129]

Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992). Class-based n-gram models of natural language. *Computational Linguistics*, 18(4). [111]

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2). [2]

Brown, P. F., Pietra, V. J. D., Pietra, V. J. D., and Mercer, R. L. (1991). Word-sense disambiguation using statistical methods. In *Proceedings of ACL*. [1, 3]

Bruce, R. F. and Wiebe, J. (1994). Word-sense disambiguation using decomposable models. In *Proceedings of ACL*. [3]

Burbea, J. and Rao, C. R. (1982). On the convexity of some divergence measures based on entropy functions. *IEEE Transactions on Information Theory*, 28:489–495. [95, 99]

Burchardt, A. (2006). Approaching textual entailment with LFG and FrameNet frames.

In *Proceedings of the Second PASCAL RTE Challenge Workshop*. [21]

Burchardt, A., Erk, K., and Frank, A. (2005). A WordNet detour to FrameNet. In *Sprachtechnologie, mobile Kommunikation und linguistische Resourcen*. [61]

Burchardt, A., Erk, K., Frank, A., Kowalski, A., Pado, S., and Pinkal, M. (2006). The SALSA corpus: a german corpus resource for lexical semantics. In *Proceedings of LREC*. [128, 129]

Burchardt, A., Pennacchiotti, M., Thater, S., and Pinkal, M. (2009). Assessing the impact of frame semantics on textual entailment. *Natural Language Engineering*, 15. [21]

Callison-Burch, C. (2008). Syntactic constraints on paraphrases extracted from parallel corpora. In *Proceedings of EMNLP*. [14]

Callison-Burch, C., Koehn, P., and Osborne, M. (2006). Improved statistical machine translation using paraphrases. In *Proceedings HLT-NAACL*. [8, 34]

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Jr., E. R. H., and Mitchell, T. M. (2010). Toward an architecture for never-ending language learning. In *Proceedings of AAAI*. [132]

Carreras, X. and Màrquez, L. (2004). Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of CoNLL*. [16]

Carreras, X. and Màrquez, L. (2005). Introduction to the CoNLL-2005 shared task: semantic role labeling. In *Proceedings of CoNLL*. [16]

Chang, M.-W., Goldwasser, D., Roth, D., and Srikumar, V. (2010). Discriminative learning over constrained latent representations. In *Proceedings of NAACL*. [15, 35, 45, 49, 125]

Chang, Y.-W. and Collins, M. (2011). Exact decoding of phrase-based translation models through Lagrangian relaxation. In *Proceedings of EMNLP*. [75]

Chapelle, O., Schölkopf, B., and Zien, A., editors (2006). *Semi-Supervised Learning*. MIT Press. [94]

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proceedings of NAACL*. [2, 4]

Chen, D., Schneider, N., Das, D., and Smith, N. A. (2010). Semafor: Frame argument resolution with log-linear models. In *Proceedings of SemEval*. [66]

Clarke, J., Goldwasser, D., Chang, M.-W., and Roth, D. (2010). Driving semantic parsing from the world's response. In *Proceedings of CoNLL*, Uppsala, Sweden. [23]

Cohn, T. and Blunsom, P. (2005). Semantic role labelling with tree conditional random fields. In *Proceedings of CoNLL*. [3]

Collins, M. (2003). Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4). [2, 3]

Corduneanu, A. and Jaakkola, T. (2003). On information regularization. In *Proceedings of UAI*. [95]

Corley, C. and Mihalcea, R. (2005). Measuring the semantic similarity of texts. In *Proceedings of ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*. [1, 15, 34]

Cover, T. M. and Thomas, J. A. (1991). *Elements of information theory*. Wiley-Interscience. [99, 104]

Culotta, A. and Sorensen, J. (2004). Dependency tree kernels for relation extraction. In *Proceedings of ACL*. [4]

Dagan, I., Glickman, O., and Magnini, B. (2005). The PASCAL recognising textual entailment challenge. In *Proceedings of MLCW*. [1, 11]

Das, D., Kumar, M., and Rudnicky, A. I. (2008). Automatic extraction of briefing templates. In *Proceedings of IJCNLP*. [7]

Das, D., Martins, A. F. T., and Smith, N. A. (2012). An exact dual decomposition algorithm for shallow semantic parsing with constraints. In *Proceedings of *SEM*. [ii, 10, 50]

Das, D. and Petrov, S. (2011). Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of ACL*. [94, 95, 101, 105, 108, 130]

Das, D., Schneider, N., Chen, D., and Smith, N. A. (2010). Probabilistic frame-semantic parsing. In *Proceedings of NAACL-HLT*. [ii, 10, 50, 70]

Das, D. and Smith, N. A. (2009). Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of ACL-IJCNLP*. [ii, 9, 34]

Das, D. and Smith, N. A. (2011). Semi-supervised frame-semantic parsing for unknown predicates. In *Proceedings of ACL*. [ii, 10, 94]

Das, D. and Smith, N. A. (2012). Graph-based lexicon expansion with sparsity-inducing penalties. In *Proceedings of NAACL-HLT*. [ii, 10, 94]

de Marneffe, M.-C., Rafferty, A. N., and Manning, C. D. (2008). Finding contradictions in text. In *Proceedings of ACL*. [8]

Dean, J. and Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1). [29, 104]

DeNero, J. and Macherey, K. (2011). Model-based aligner combination using dual decomposition. In *Proceedings of ACL*. [75]

Deschacht, K. and Moens, M.-F. (2009). Semi-supervised semantic role labeling using the Latent Words Language Model. In *Proceedings of EMNLP*. [22]

Dhillon, P. S., Foster, D., and Ungar, L. (2011). Multi-view learning of word embeddings via cca. In *Proc. of NIPS*. [94]

Dolan, B., Quirk, C., and Brockett, C. (2004). Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources. In *Proceedings of COLING*. [14, 43]

Dolan, W. B. and Brockett, C. (2005). Automatically constructing a corpus of sentential paraphrases. In *Proceedings of IWP*. [7, 13, 43, 45, 127]

Duchi, J. C., Shalev-Shwartz, S., Singer, Y., and Chandra, T. (2008). Efficient projections onto the l_1-ball for learning in high dimensions. In *Proceedings of ICML*. [83]

Eisner, J. (1996). Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING*. [25]

Erk, K. and Padó, S. (2006). Shalmaneser - a toolchain for shallow semantic parsing. In *Proceedings of LREC*. [20]

Fellbaum, C., editor (1998). *WordNet: an electronic lexical database*. [7, 31, 45, 56]

Ferrucci, D., Nyberg, E., Allen, J., Barker, K., Brown, E. W., Chu-Carroll, J., Ciccolo, A., Duboue, P. A., Fan, J., Gondek, D., Hovy, E., Katz, B., Lally, A., McCord, M., Morarescu, P., Murdock, J. W., Porter, B., Prager, J. M., Strzalkowski, T., Welty, C., and Zadrozny, W. (2008). Towards the open advancement of question answering systems. Technical Report RC24789, IBM Research. [24]

Fillmore, C. J. (1982). Frame Semantics. In *Linguistics in the Morning Calm*. [i, 4, 11, 19, 50]

Fillmore, C. J., Johnson, C. R., and Petruck, M. R. (2003). Background to FrameNet. *International Journal of Lexicography*, 16(3). [5, 19, 50, 111]

Finch, A., Hwang, Y. S., and Sumita, E. (2005). Using machine translation evaluation techniques to determine sentence-level semantic equivalence. In *Proceedings of IWP*. [15, 34]

Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of ACL*. [3]

Fleischman, M., Kwon, N., and Hovy, E. (2003). Maximum entropy models for FrameNet classification. In *Proceedings of EMNLP*. [20]

Fung, P. and Chen, B. (2004). BiFrameNet: bilingual frame semantics resource construction by cross-lingual induction. In *Proceedings of COLING*. [21]

Fürstenau, H. and Lapata, M. (2009a). Graph alignment for semi-supervised semantic role labeling. In *Proceedings of EMNLP*. [21, 22]

Fürstenau, H. and Lapata, M. (2009b). Semi-supervised semantic role labeling. In *Proceedings of EACL*. [21, 22]

Ge, R. and Mooney, R. J. (2005). A statistical semantic parser that integrates syntax and semantics. In *Proceedings of CoNLL*. [1, 23]

Gerber, M. and Chai, J. Y. (2010). Beyond NomBank: A study of implicit arguments for nominal predicates. In *Proceedings of ACL*. [78]

Giampiccolo, D., Magnini, B., Dagan, I., and Dolan, B. (2007). The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*. [1, 11]

Gildea, D. and Jurafsky, D. (2002). Automatic labeling of semantic roles. *Computational Linguistics*, 28(3). [3, 16, 20]

Gimpel, K. and Smith, N. A. (2009). Feature-rich translation by quasi-synchronous lattice parsing. In *Proceedings of EMNLP*. [36]

Gimpel, K. and Smith, N. A. (2011). Quasi-synchronous phrase dependency grammars for machine translation. In *Proceedings of EMNLP*. [36]

Giuglea, A. and Moschitti, A. (2006). Shallow semantic parsing based on FrameNet, VerbNet and PropBank. In *Proceedings of ECAI 2006*. [20, 21]

Glickman, O. and Dagan, I. (2005). Web based probabilistic textual entailment. In *Proceedings of PASCAL Challenge Workshop for Recognizing Textual Entailmen*. [3, 12]

Glickman, O., Dagan, I., and Koppel, M. (2005). A probabilistic classification approach for lexical textual entailment. In *Proceedings of AAAI*. [1, 3]

Graff, D. (2003). English Gigaword. Linguistic Data Consortium. [40, 106, 111]

Grandvalet, Y. and Bengio, Y. (2004). Semi-supervised learning by entropy minimization. In *Proceedings of NIPS*. [101]

Grishman, R. and Sundheim, B. (1995). Design of the MUC-6 evaluation. In *Proceedings of MUC*. [2]

Gropp, W., Lusk, E., and Skjellum, A. (1994). *Using MPI: Portable Parallel Programming with the Message-Passing Interface*. MIT Press. [30, 62, 70, 104]

Haghighi, A., Liang, P., Berg-Kirkpatrick, T., and Klein, D. (2008). Learning bilingual lexicons from monolingual corpora. In *Proceedings of ACL-HLT*. [3]

Hajič, J., Ciaramita, M., Johansson, R., Kawahara, D., Martí, M. A., Màrquez, L., Meyers, A., Nivre, J., Padó, S., Štěpánek, J., Straňák, P., Surdeanu, M., Xue, N., and Zhang, Y. (2009). The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL*. [19]

Hall, K., McDonald, R. T., Katz-Brown, J., and Ringgaard, M. (2011). Training dependency parsers by jointly optimizing multiple objectives. In *Proceedings of EMNLP*. [89]

Harabagiu, S. and Hickl, A. (2006). Methods for using textual entailment in open-domain question answering. In *Proceedings of COLING-ACL*. [8]

Heilman, M. and Smith, N. A. (2010). Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proceedings of NAACL-HLT*. [15]

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800. [35]

Huang, F. and Yates, A. (2009). Distributional representations for handling sparsity in supervised sequence labeling. In *Proc. of ACL*. [94]

Jiao, F., Wang, S., Lee, C.-H., Greiner, R., and Schuurmans, D. (2006). Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Pro-*

ceedings of ACL. [94]

Jijkoun, V. and de Rijke, M. (2005). Recognizing textual entailment using lexical similarity. In *Proceedings of the PASCAL Challenges Workshop on Recognizing Textual Entailment*. [12]

Joachims, T. (1999). Making large-scale SVM learning practical. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press. [45]

Johansson, R. and Nugues, P. (2007). LTH: semantic structure extraction using nonprojective dependency trees. In *Proceedings of SemEval*. [20, 57, 58, 61, 79]

Johansson, R. and Nugues, P. (2008). Dependency-based semantic role labeling of PropBank. In *Proceedings of EMNLP*. [56]

Kate, R. J., Wong, Y. W., and Mooney, R. J. (2005). Learning to transform natural to formal languages. In *Proceedings of AAAI*. [23]

Kingsbury, P. and Palmer, M. (2002). From TreeBank to PropBank. In *Proceedings of LREC*. [7, 16, 17, 74]

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL*. [3]

Kok, S. and Brockett, C. (2010). Hitting the right paraphrases in good time. In *Proceedings of NAACL-HLT*. [14]

Komodakis, N., Paragios, N., and Tziritas, G. (2007). MRF optimization via dual decomposition: Message-passing revisited. In *ICCV*. [74, 80]

Koo, T., Rush, A. M., Collins, M., Jaakkola, T., and Sontag, D. (2010). Dual decomposition for parsing with non-projective head automata. In *Proceedings of EMNLP*. [74]

Kowalski, M. and Torrésani, B. (2009). Sparsity and persistence: mixed norms provide simple signal models with dependent coefficients. *Signal, Image and Video Processing*, 3:251–264. [96, 101]

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *Annals of Mathematical Statistics*, 22. [99]

Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., and Steedman, M. (2011). Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of EMNLP*. [22]

Liang, P., Jordan, M. I., and Klein, D. (2011). Learning dependency-based compositional semantics. In *Proceedings of ACL*. [1, 23]

Lin, D. (1993). Principle-based parsing without overgeneration. In *Proceedings of ACL*. [111]

Lin, D. (1994). Principar–an efficient, broadcoverage, principle-based parser. In *Proceedings of COLING*. [111]

Lin, D. (1998). Automatic retrieval and clustering of similar words. In *Proceedings of COLING-ACL*. [111]

Lin, D. and Pantel, P. (2001). DIRT - discovery of inference rules from text. In *Proceedings of KDD*. [13]

Lin, D. and Wu, X. (2009). Phrase clustering for discriminative learning. In *Proceedings of ACL-IJCNLP*. [13]

Lin, J. (1991). Divergence measures based on the Shannon entropy. *IEEE Transactions on Information theory*, 37:145–151. [95, 99]

Liu, D. and Gildea, D. (2010). Semantic role features for machine translation. In *Proceedings of COLING*. [7]

Liu, D. C. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Math. Programming*, 45(3). [28, 30, 70]

MacCartney, B., Grenager, T., de Marneffe, M.-C., Cer, D., and Manning, C. D. (2006). Learning to recognize features of valid textual entailments. In *Proceedings of HLT-NAACL*. [12]

MacCartney, B. and Manning, C. D. (2007). Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*. [1, 12, 47]

Malakasiotis, P. (2009). Paraphrase recognition using machine learning to combine similarity measures. In *Proceedings of ACL Student Research Workshop*. [15]

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics*, 19(2):313–330. [2, 3, 16, 17, 96, 106]

Màrquez, L., Carreras, X., Litkowski, K. C., and Stevenson, S. (2008). Semantic role labeling: an introduction to the special issue. *Computational Linguistics*, 34(2). [19, 74, 75]

Marsi, E. and Krahmer, E. (2005). Explorations in sentence fusion. In *Proceedings of EWNLG*. [34]

Martins, A. F. T., Das, D., Smith, N. A., and Xing, E. P. (2008). Stacking dependency parsers. In *Proceedings of EMNLP*. [4, 26, 32]

Martins, A. F. T., Figueiredo, M. A. T., Aguiar, P. M. Q., Smith, N. A., and Xing, E. P. (2011a). An augmented Lagrangian approach to constrained MAP inference. In *Proceedings of ICML*. [74, 80]

Martins, A. F. T., Smith, N. A., Aguiar, P. M. Q., and Figueiredo, M. A. T. (2011b). Dual decomposition with many overlapping components. In *Proceedings of EMNLP*. [75, 80, 81, 83, 84, 85, 86]

Martins, A. F. T., Smith, N. A., and Xing, E. P. (2009). Concise integer linear programming formulations for dependency parsing. In *Proceedings of ACL-IJCNLP*. [74]

Martins, A. F. T., Smith, N. A., Xing, E. P., Figueiredo, M. A. T., and Aguiar, P. M. Q. (2010). Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of EMNLP*. [74, 79, 80]

Marton, Y., Callison-Burch, C., and Resnik, P. (2009). Improved statistical machine translation using monolingually-derived paraphrases. In *Proceedings of EMNLP*. [8]

Matsubayashi, Y., Okazaki, N., and Tsujii, J. (2009). A comparative study on generalization of semantic roles in FrameNet. In *Proceedings of ACL-IJCNLP*. [21, 67]

McClosky, D., Charniak, E., and Johnson, M. (2006). Effective self-training for parsing. In *Proceedings of HLT-NAACL*. [94]

McDonald, R., Crammer, K., and Pereira, F. (2005). Online large-margin training of dependency parsers. In *Proceedings of ACL*. [4, 26, 32, 45]

McKeown, K. R. (1979). Paraphrasing using given and new information in a question-answer system. In *Proceedings of ACL*. [13]

Melamed, I. D. (2004). Statistical machine translation by parsing. In *Proceedings of ACL*.

[37]

Merialdo, B. (1994). Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2). [94]

Meyers, A., Reeves, R., Macleod, C., Szekely, R., Zielinska, V., Young, B., and Grishman, R. (2004). The NomBank project: An interim report. In *Proceedings of NAACL/HLT Workshop on Frontiers in Corpus Annotation*. [20, 74]

Moschitti, A., Morărescu, P., and Harabagiu, S. M. (2003). Open-domain information extraction via automatic semantic labeling. In *Proceedings of FLAIRS*. [21]

Narayanan, S. and Harabagiu, S. (2004). Question answering based on semantic structures. In *Proceedings of COLING*. [21]

Nivre, J., Hall, J., and Nilsson, J. (2004). Memory-based dependency parsing. In *Proceedings of CoNLL*. [4]

Ohara, K. H. (2011). Full text annotations in japanese FrameNet: Annotating BCCWJ with semantic frames. In *Proceedings of the Annual Meeting of the Association for Natural Language Processing*. [129]

O'Sullivan, J. A. (1998). Alternating minimization algorithms: from Blahut-Arimoto to Expectation-Maximization. In Vardy, A., editor, *Codes, Curves, and Signals: Common Threads in Communications*, pages 173–192. Kluwer. [99]

Padó, S. and Erk, K. (2005). To cause or not to cause: cross-lingual semantic matching for paraphrase modelling. In *Proceedings of the Cross-Language Knowledge Induction Workshop*. [21]

Padó, S., Galley, M., Jurafsky, D., and Manning, C. D. (2009). Robust machine translation evaluation with entailment features. In *Proceedings of ACL-IJCNLP*. [8]

Padó, S. and Lapata, M. (2005a). Cross-lingual bootstrapping for semantic lexicons: The case of FrameNet. In *Proceedings of AAAI*. [129]

Padó, S. and Lapata, M. (2005b). Cross-linguistic projection of role-semantic information. In *Proceedings of HLT-EMNLP*. [21]

Palmer, M., Gildea, D., and Kingsbury, P. (2005). The Proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1). [1]

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2001). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL*. [7, 8, 45]

Pennacchiotti, M., Cao, D. D., Basili, R., Croce, D., and Roth, M. (2008). Automatic induction of FrameNet lexical units. In *Proceedings of EMNLP*. [21]

Pereira, F., Tishby, N., and Lee, L. (1993). Distributional clustering of English words. In *Proceedings of ACL*. [13]

Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *Proceedings of COLING/ACL*. [3]

Petrov, S. and Klein, D. (2008). Sparse multi-scale grammars for discriminative latent variable parsing. In *Proceedings of EMNLP*. [3, 28, 29, 62]

Pradhan, S. S., Ward, W. H., Hacioglu, K., Martin, J. H., and Jurafsky, D. (2004). Shallow semantic parsing using support vector machines. In *Proceedings of HLT-NAACL*. [19]

Punyakanok, V., Roth, D., W.-T. Yih, and Zimak, D. (2004). Semantic role labeling via integer linear programming inference. In *Proceedings of COLING*. [19, 74, 76, 78]

Qiu, L., Kan, M.-Y., and Chua, T.-S. (2006). Paraphrase recognition via dissimilarity significance classification. In *Proceedings of EMNLP*. [7, 15, 128]

Quirk, C., Brockett, C., and Dolan, W. B. (2004). Monolingual machine translation for paraphrase generation. In *Proceedings of EMNLP*. [14, 43]

Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP*. [39, 45, 56]

Ravichandran, D. and Hovy, E. (2002). Learning surface text patterns for a question answering system. In *Proceedings of ACL*. [13]

Roth, D. and Yih, W. (2004). A linear programming formulation for global inference in natural language tasks. In *Proceedings of CoNLL*. [74]

Ruppenhofer, J., Ellsworth, M., Petruck, M. R. L., Johnson, C. R., and Scheffczyk, J. (2006). FrameNet II: extended theory and practice. [71]

Rush, A. M. and Collins, M. (2011). Exact decoding of syntactic translation models through Lagrangian relaxation. In *Proceedings of ACL*. [75]

Rush, A. M., Sontag, D., Collins, M., and Jaakkola, T. (2010). On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of EMNLP*. [74, 80, 81]

Salomão, M. M. (2009). Framenet brasil: um trabalho em progresso. *Calidoscṕpio*, 7(3). [129]

Schuler, K. K. (2005). *VerbNet: a broad-coverage, comprehensive verb lexicon*. PhD thesis, University of Pennsylvania. [19, 90]

Shen, D. and Lapata, M. (2007). Using semantic roles to improve question answering. In *Proceedings of EMNLP-CoNLL*. [7, 21]

Shi, L. and Mihalcea, R. (2004). An algorithm for open text semantic parsing. In *Proceedings of Workshop on Robust Methods in Analysis of Natural Language Data*. [20]

Shi, L. and Mihalcea, R. (2005). Putting pieces together: combining FrameNet, VerbNet and WordNet for robust semantic parsing. In *Computational Linguistics and Intelligent Text Processing*. [21]

Smith, D. and Eisner, J. (2008). Dependency parsing by belief propagation. In *Proceedings of EMNLP*. [79, 80]

Smith, D. A. and Eisner, J. (2006). Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings of the HLT-NAACL Workshop on Statistical Machine Translation*. [15, 34, 35, 36, 37, 39]

Smith, D. A. and Eisner, J. (2007). Bootstrapping feature-rich dependency parsers with entropic priors. In *Proceedings of EMNLP*. [94]

Smith, D. A. and Eisner, J. (2009). Parser adaptation and projection with quasi-synchronous grammar features. In *Proceedings of EMNLP*. [36]

Smith, D. A. and Smith, N. A. (2007). Probabilistic models of nonprojective dependency trees. In *Proceedings of EMNLP-CoNLL*. [3]

Smith, N. A. (2006). *Novel estimation methods for unsupervised discovery of latent structure in natural language text*. PhD thesis, Johns Hopkins University. [3]

Smith, N. A. and Eisner, J. (2005). Contrastive estimation: training log-linear models on unlabeled data. In *Proceedings of ACL*. [94]

Snow, R., Jurafsky, D., and Ng, A. Y. (2006). Semantic taxonomy induction from heterogenous evidence. In *Proceedings of COLING-ACL*. [3]

Socher, R., Huang, E. H., Pennington, J., Ng, A. Y., and Manning, C. D. (2011). Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceed-*

ings of NIPS. [16, 35, 45, 48, 49, 125]

Srikumar, V. and Roth, D. (2011). A joint model for extended semantic role labeling. In *Proceedings of EMNLP*. [90]

Steedman, M. (1996). *Surface Structure and Interpretation*. MIT Press. [22]

Steedman, M. (2000). *The Syntactic Process*. MIT Press. [22]

Subirats, C. (2004). Framenet español. una red semántica de marcos conceptuales. In *Proceedings of International Congress of Hispanic Linguistics*. [129]

Subramanya, A. and Bilmes, J. (2008). Soft-supervised learning for text classification. In *Proceedings of EMNLP*. [95, 99, 100, 101, 104]

Subramanya, A. and Bilmes, J. (2009). Entropic graph regularization in non-parametric semi-supervised classification. In *Proceedings of NIPS*. [95, 96, 100, 101, 104, 105, 123, 124]

Subramanya, A., Petrov, S., and Pereira, F. (2010). Efficient graph-based semi-supervised learning of structured tagging models. In *Proceedings of EMNLP*. [95, 101, 103, 105, 108]

Surdeanu, M., Harabagiu, S., Williams, J., and Aarseth, P. (2003). Using predicate-argument structures for information extraction. In *Proceedings of ACL*. [21]

Surdeanu, M., Johansson, R., Meyers, A., Màrquez, L., and Nivre, J. (2008). The CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of CoNLL*. [22]

Szummer, M. and Jaakkola, T. (2001). Partially labeled classification with Markov random walks. In *Proceedings of NIPS*. [95]

Talukdar, P. P. (2010). *Graph-Based Weakly-Supervised Methods for Information Extraction and Integration*. PhD thesis, University of Pennsylvania. [101]

Talukdar, P. P. and Crammer, K. (2009). New regularized algorithms for transductive learning. In *Proceedings of the ECML-PKDD*. [95]

Thompson, C. A., Levy, R., and Manning, C. D. (2003). A generative model for semantic role labeling. In *Proceedings of ECML*. [20]

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B)*, 58:267–288. [101]

Tjong Kim Sang, E. F. (2002). Introduction to the CoNLL-2002 shared task: language-independent named entity recognition. In *Proceedings of CoNLL*. [2]

Tjong Kim Sang, E. F. and Buchholz, S. (2000). Introduction to the CoNLL-2000 shared task: chunking. In *Proceedings of CoNLL*. [2]

Toutanova, K., Haghighi, A., and Manning, C. (2005). Joint learning improves semantic role labeling. In *Proceedings of ACL*. [19, 71, 76]

Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning. In *Proc. of ACL*. [94]

Turian, J. P., Shen, L., and Melamed, I. D. (2003). Evaluation of machine translation and its evaluation. In *Proceedings of Machine Translation Summit IX*. [45]

Vapnik, V. N. (1995). *The nature of statistical learning theory*. Springer. [15]

Wan, S., Dras, M., Dale, R., and Paris, C. (2006). Using dependency-based features to take the "para-farce" out of paraphrase. In *Proceedings of ALTW*. [15, 34, 35, 45]

Wang, M., Smith, N. A., and Mitamura, T. (2007). What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proceedings of EMNLP-CoNLL*. [4, 7, 29, 34, 36, 37, 39, 40]

Weston, J., Ratle, F., and Collobert, R. (2008). Deep learning via semi-supervised embedding. In *Proceedings of ICML*. [22]

White, T. (2009). *Hadoop: The Definitive Guide*. O'Reilly Media. [30]

Wong, Y. W. and Mooney, R. J. (2006). Learning for semantic parsing with statistical machine translation. In *Proceedings of HLT-NAACL*. [23]

Wong, Y. W. and Mooney, R. J. (2007). Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of ACL*. [23]

Wu, D. (1997). Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3). [15]

Wu, D. (2005). Recognizing paraphrases and textual entailment using inversion transduction grammars. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*. [15]

Wu, D. and Fung, P. (2009). Semantic roles for SMT: a hybrid two-pass model. In *Proceedings of NAACL*. [7]

Yamada, H. and Matsumoto, Y. (2003). Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*. [26, 38]

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of ACL*. [1, 3]

Yi, S.-T., Loper, E., and Palmer, M. (2007). Can semantic roles generalize across genres? In *Proceedings of HLT-NAACL*. [1, 19]

Zelle, J. M. and Mooney, R. J. (1996). Learning to parse database queries using inductive logic programming. In *Proceedings of AAAI/IAAI*. [23]

Zettlemoyer, L. S. and Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of UAI*. [1, 22]

Zettlemoyer, L. S. and Collins, M. (2007). Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of EMNLP-CoNLL*. [22]

Zhang, Y. and Patrick, J. (2005). Paraphrase identification by text canonicalization. In *Proceedings of ALTW*. [14, 34]

Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J. (1997). Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software*, 23:550–560. [96, 104]

Zhu, X. (2008). Semi-Supervised Learning Literature Survey. Online publication. [94]

Zhu, X. and Ghahramani, Z. (2002). Learning from labeled and unlabeled data with label propagation. Technical report, Carnegie Mellon University. [95, 96]

Zhu, X., Ghahramani, Z., and Lafferty, J. (2003). Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of ICML*. [95, 99, 104, 124]

Zollmann, A. and Venugopal, A. (2006). Syntax augmented machine translation via chart parsing. In *Proceedings of WMT*. [4]

Appendix A

Test Set for FrameNet 1.5 Release

Name
ANC__110CYL067
ANC__110CYL069
ANC__112C-L013
ANC__IntroHongKong
ANC__StephanopoulosCrimes
ANC__WhereToHongKong
KBEval__atm
KBEval__Brandeis
KBEval__cycorp
KBEval__parc
KBEval__Stanford
KBEval__utd-icsi
LUCorpus-v0.3__20000410_nyt-NEW
LUCorpus-v0.3__AFGP-2002-602187-Trans
LUCorpus-v0.3__enron-thread-159550
LUCorpus-v0.3__IZ-060316-01-Trans-1
LUCorpus-v0.3__SNO-525
LUCorpus-v0.3__sw2025-ms98-a-trans.ascii-1-NEW
Miscellaneous__Hound-Ch14
Miscellaneous__SadatAssassination
NTI__NorthKorea_Introduction
NTI__Syria_NuclearOverview
PropBank__AetnaLifeAndCasualty

Table A.1: Names of the documents in our test set, taken from the full-text section of the FrameNet 1.5 release.

We include the names of the test documents in Table A.1 to facilitate fair replication of our work. The test set contains a mix of several sources from which these documents were drawn. The FrameNet 1.5 release, whose description is given in §5.1.2 contained 78 documents. We chose the 23 documents mentioned in the table for testing the performance of our statistical models. These contained a total of 4,458 annotated targets.

Vita

Dipanjan Das was born in Calcutta, India in 1983. After graduating from the M.P. Birla Foundation Higher Secondary School, Calcutta in 2001, he completed a Bachelor of Technology in Computer Science and Engineering (with honors) from the Indian Institute of Technology, Kharagpur. At IIT, his senior project on text chunking of free word order languages was awarded the best undergraduate thesis in Computer Science and Engineering. Dipanjan was also awarded the Dr. B.C. Roy Memorial Gold Medal for best all-round performance in academics and co-curricular activities among the graduating class of 2005. After spending a year at the Department of Computer Science of SUNY, Stony Brook, Dipanjan started his graduate study at the Language Technologies Institute, School of Computer Science at Carnegie Mellon University, where he earned an M.S. degree in August 2008, working on language generation. He earned his Ph.D. degree from the same institute in May 2012, with a dissertation on computational semantics. His research spans several areas of natural language processing, including unsupervised and semi-supervised learning of linguistic structure. In 2011, his work on multilingual learning of syntax won the best paper award at the 49th Annual Meeting of the Association for Computational Linguistics. Dipanjan will join Google, New York as a research scientist in June 2012, and can be found on the web at `http://www.dipanjandas.com`.