Structured Probabilistic Models for Natural Language Semantics

Dipanjan Das

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue,
Pittsburgh, PA 15213, USA

April 23, 2010

Dissertation Committee:
Noah A. Smith (chair), Carnegie Mellon University
William W. Cohen, Carnegie Mellon University
Lori S. Levin, Carnegie Mellon University
Dan Roth, University of Illinois, Urbana-Champaign

Abstract

The recent past has witnessed a predominance of robust empirical methods in natural language structure prediction, but mostly through the analysis of *syntax*. Wide-coverage analysis of the underlying meaning or *semantics* of natural language utterances still remains a major obstacle for language understanding. A primary bottleneck 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 thesis proposal, we study structured probabilistic models tailored to solve problems in computational semantics, with a focus on modeling structure that is not visible in annotated text data.

First, we investigate the problem of **paraphrase identification**, which attempts to recognize whether two sentences convey the same meaning. Our approach towards solving this problem systematically blends natural language syntax and lexical semantics within a probabilistic framework, and achieves state-of-the-art accuracy on a standard corpus, when trained on a set of true and false sentential paraphrases (Das and Smith, 2009). Given a pair of sentences, the presented model recognizes the paraphrase relationship by predicting the structure of one sentence given the other, allowing loose syntactic transformation and lexical semantic alteration at the level of aligned words.

Second, we focus on the problem of **frame-semantic parsing**. Frame semantics offers deep linguistic analysis that exploits the use of lexical semantic properties and relationships among semantic frames and roles. We describe probabilistic models for analyzing a sentence to produce a full frame-semantic parse. Our models leverage the FrameNet (Fillmore et al., 2003) and WordNet (Fellbaum, 1998) lexica, a small corpus containing full text annotations of natural language sentences, syntactic representations from which we derive features, and results in significant improvements over previously published results on the same corpus (Das et al., 2010a).

Unfortunately, the datasets used to train our paraphrase and frame-semantic models are too small to lead to robust performance. Therefore, to obviate this problem, 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. For the paraphrase problem, our model assumes the presence of hidden alignments between the syntactic structures of the sentence pair, which while unknown during the training and testing phases, produce meaningful correpondence between the two sentences' syntax at inference time, as a by-product. For frame-semantic parsing, we face the challenge of identifying semantic frames for previously unseen lexical items. To generalize our model to these new lexical items, we adopt a stochastic process that assumes that a given semantic frame generates a latent lexical item, and the lexical item in turn generates the unseen item through a lexical semantic transformation process.

Continuing with the theme of hypothesizing hidden structure in data for modeling of natural language semantics, we propose to leverage large volumes of unlabeled data to improve upon the aforementioned tasks. As an attempt to harvest raw data to boost semantic structure prediction, we intend to gather categorical and topical word clusters from a large corpus using standard clustering techniques that look at lexical and syntactic contexts around a given word. Inspired by recent advances in syntactic parsing (Koo et al., 2008) and information extraction (Miller et al., 2004; Lin and Wu, 2009), we propose to incorporate features based on these clusters in our existing models for paraphrase identification and frame-semantic parsing, hoping to resolve data sparsity to some extent

and in turn help improve the performance of our models in the tasks' respective evaluation metrics. For the task of frame-semantic parsing, we next propose to use phrasal paraphrases to improve the subtask of argument identification of semantic frames. The miniscule size of the annotated corpus for this subtask offers very few labeled phrases as examples for each frame element (or semantic role) in the FrameNet lexicon. We propose to use phrasal paraphrases extracted from unlabeled corpora through state-of-the-art methods (Callison-Burch, 2008) to alleviate this problem, and improve argument identification by incorporating paraphrase information as features in our models. The two tasks of modeling paraphrase and frame-semantic parsing shed light on two different parts of the bigger challenge of modeling semantics; however, using similar methods and features for both problems will help us move toward a more unified representation.

Finally, we propose to investigate **semi-supervised learning** techniques to improve frame-semantic parsing. Recent work on semi-supervised learning for language processing has focused on various techniques, often resulting in significant improvements over supervised methods. Two examples are semi-supervised structured conditional models (Suzuki and Isozaki, 2008; Suzuki et al., 2009) and entropy minimization (Jiao et al., 2006; Smith and Eisner, 2007, *inter alia*). The former extends a standard set of features whose weights are estimated from labeled data, with another feature set induced from unlabeled data, and is trained by alternating parameter estimation on the annotated data and the unlabeled data till convergence. Bootstrapping feature rich probabilistic models with the latter is another avenue for leveraging unlabeled data, and has exhibited improvements for sequence labeling tasks and dependency parsing. Frame-semantic parsing naturally fits into either semi-supervised learning framework because available annotated data is scarce, and our modeling techniques are amenable to semi-supervised extensions.

This dissertation will empirically demonstrate that unannotated 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.

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 **logical forms** using training corpora belonging to a narrow domain (Ge and Mooney, 2005; Zettlemoyer and Collins, 2005), 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 limited set of shallow tags to gather sufficient amount of training data, resulting in inconsistency across different semantic frames (Yi et al., 2007). **Word sense diambiguation** 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 can prove to be 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 have 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) either because these models are trained on restricted domain corpora or use inference mechanisms that do not capture complex natural language phenomena.

In this dissertation proposal, we investigate structured probabilistic 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 all 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 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 an-

notated data, we model **latent structure** in data for both the tasks in consideration. Finally, we propose to use large volumes of **unlabeled data** to derive features for our models and perform semi-supervised learning to demonstrate that useful semantic information for analysis can be harvested 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), corefence resolution (Grishman and Sundheim, 1995) and machine translation (Al-Onaizan et al., 1999), 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. For example, to cite a few, these models facilitate the combination of simpler models, promote the use of ovelapping 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 and Klein, 2008; Smith and Smith, 2007), 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 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.

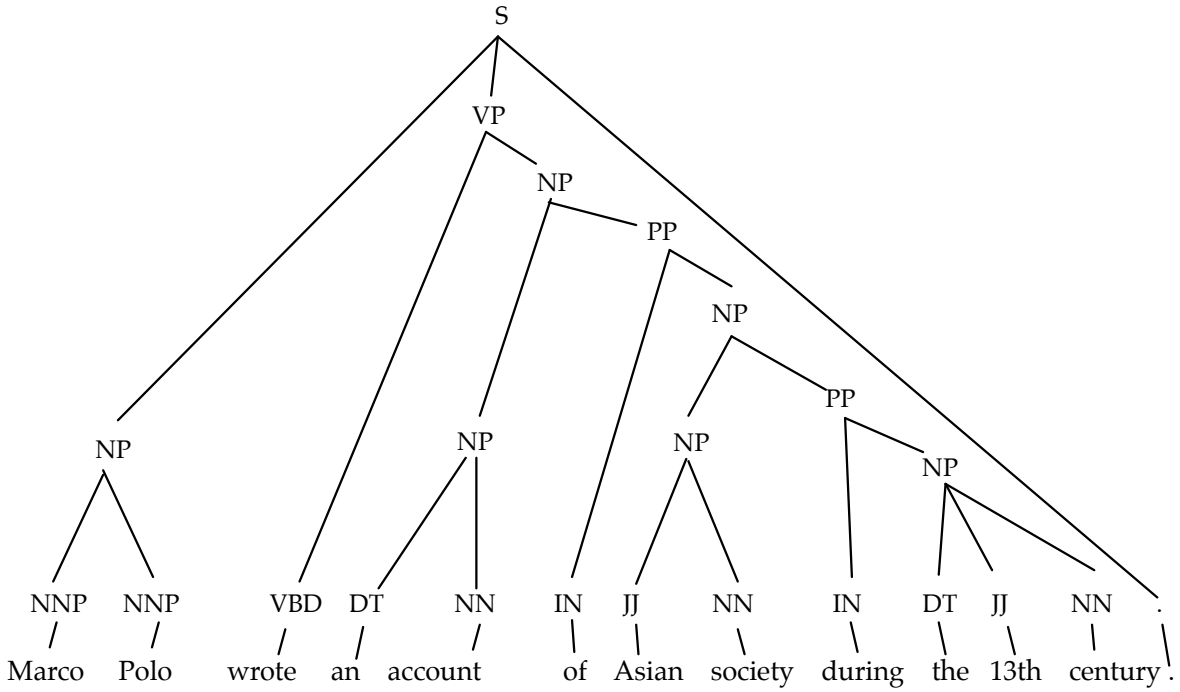

Figure 1.1: A Penn Treebank style phrase-structure syntax tree for Example 1.1.

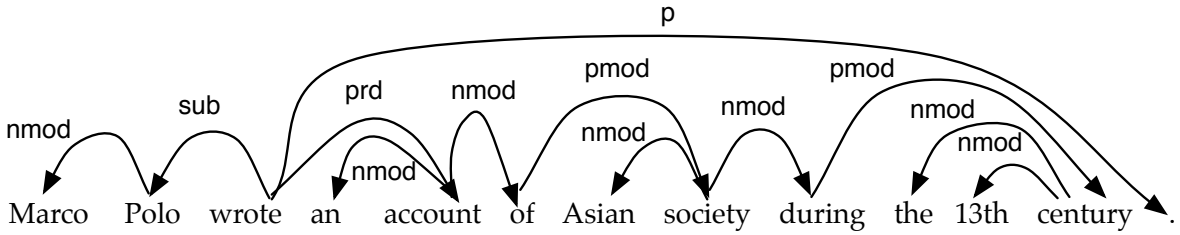

Figure 1.2: A labeled dependency syntax tree for Example 1.1.

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 would lead to representations beyond formalisms such as dependency and phrase-structure grammars that model syntax. 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 (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

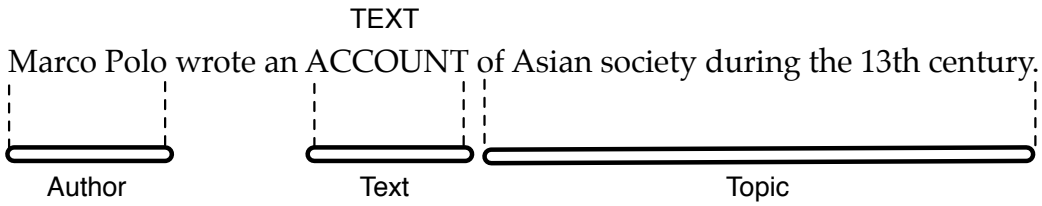

Figure 1.3: A frame-semantic parse for Example 1.1.

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 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.

What applications can benefit from deep 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 following:

(1.2) मार्को पोलो १३ बीं सदी के दौरान एशियाइ समाज के एक खाते में लिखा था।

In this sentence, 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.

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

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

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 एशियाइ समाज का इतिहास लिखा था।

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 incorporate 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) style 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 style 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).

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 (two sentences conveying the same information), and contradiction (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. 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 aid that.

1.3 Contributions of the Thesis

As described at the onset of this chapter, we investigate probabilistic 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. The major contributions of the thesis are:

1. We model complex semantic phenomena using structured probabilistic models, instead of relying on a collection of naïve classifiers. To this end, whenever possible, we make use of distributed computing to facilitate fast parameter estimation.

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 Viterbi inference.

3. Our models provide unique direction in modeling the two semantic analysis problems and 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.

4. Finally, we propose to use vast amounts of unlabeled data for improved modeling of the aforementioned phenomena. We plan to use features derived from word clusters found in large unannotated text corpora, as well use semi-supervised learning to learn

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

models from a mixture of labeled and unlabeled data. Our probabilistic models are amenable to both these extensions and we hope to demonstrate that unlabeled data contains information that can significantly benefit the current methods.

1.4 Organization of the Document

This document is organized as follows. §2 focuses on relevant previous work on semantic relationships between sentence pairs, with a focus on modeling paraphrase. Next, it describes relevant work on shallow semantic parsing, especially FrameNet based analysis of text. Herein, we contrast our techniques with previous work, and focus on the advantages of probabilistic modeling techniques. §3 describes a set of tools used in this thesis. Examples of these tools are syntactic representations, probabilistic log-linear models and word clustering methods used for deriving features from unlabeled data. §4 investigates our model for recognizing a paraphrase relationship between two sentences. We describe our probabilistic technique, the experiments and the results achieved on a popoular corpus. §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 probabilistic model used to analyze raw text, and finally explains the experiments and the results achieved on a standard dataset. §6 consists of proposed work to be completed as a part of the final dissertation. We consider possibilities of improving the frame-semantic parser of §5 by appending our model with features derived from unlabeled corpora, as well as semi-supervised extensions. Finally, §7 concludes the findings of this research and tabulates a timeline for the dissertation.

Chapter 2

Literature Review

This chapter reviews previous 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 and §2.2 provide background material for the detailed models described in §4 for paraphrase identification and §5 for frame-semantic parsing, respectively.

2.1 Models of Sentence-sentence Relationships

In recent years, modeling semantic relationships between sentence pairs has generated considerable interest in the NLP community. Among various relationships like entailment, paraphrase and contradition, 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".

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.

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

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, 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 deep 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 is 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 stricter 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.

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.

Finding paraphrases at finer granularity levels of *words* and *phrases* have 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 extentions 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 given 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.

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, 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 our model in an elegant way. Word alignments are also modeled in our method, and are treated as latent variables and are marginalized out.

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, while Wu did not estimate any parameters from data at all.

2.2 Techniques in Shallow Semantic Parsing

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, 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 ∼40,000 sentences, thus making the semantic role labeling task an attractive one from the perspective of machine learning.

There are many instances of influential work on semantic role labeling using Propbank conventions. Pradhan et al. (2004) present a system that use SVMs to identify the arguments in a syntax tree that can serve as semantic roles, followed by the classification of the identified arguments resulting the role names. 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) is a repository for few other interesting papers on the topic, leveraging the Propbank conventions for labeling shallow semantic structures.

In this 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).

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 focuses 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 used the newer FrameNet v. 1.3,[2] that lists 795 frames and 7124 roles, 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 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, model the whole problem with only two probabilistic models, and result in significantly better overall parsing scores.

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, 2005; Fürstenau and Lapata, 2009). 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] Available at `http://framenet.icsi.berkeley.edu` as of April 9, 2010.

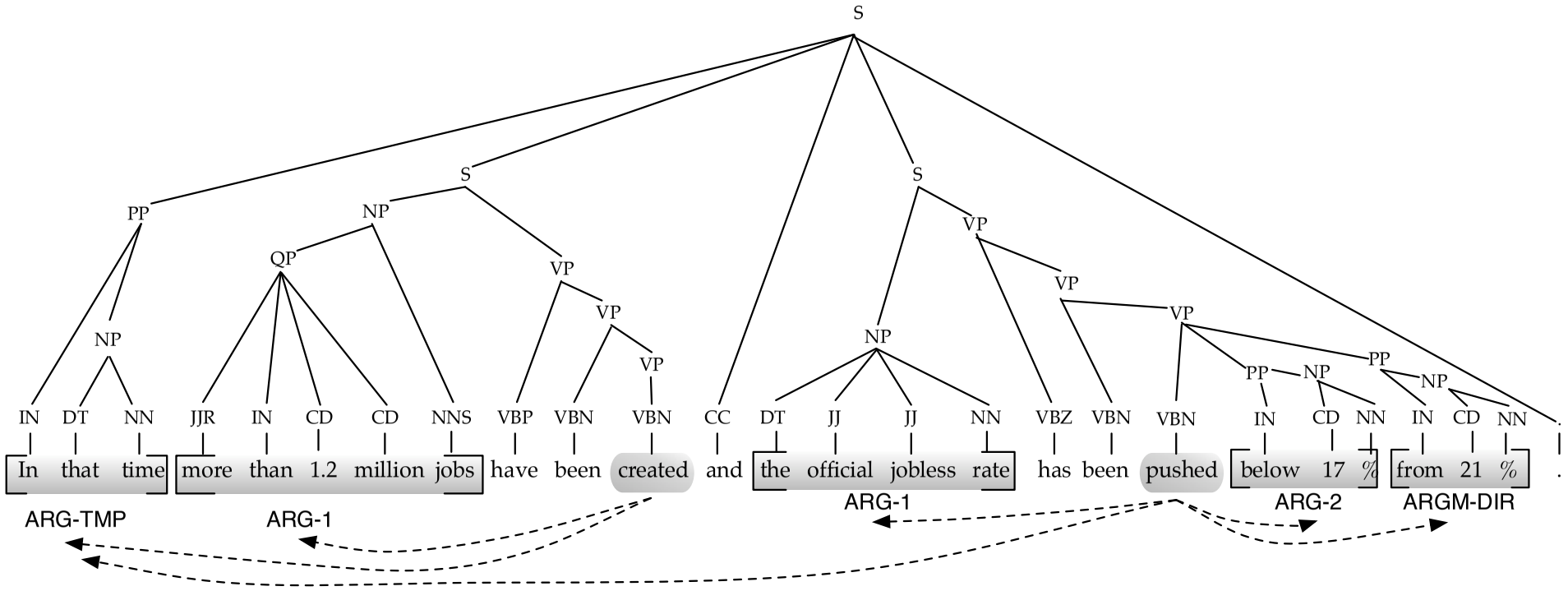

Figure 2.1: A phrase-structure tree taken from the Penn Treebank Marcus et al. (1993), and annotated with predicates and corresponding semantic roles in the PropBank (Kingsbury and Palmer, 2002). There are two verbs marked in rounded rectangles- "created" and "pushed" that serve as the predicates in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

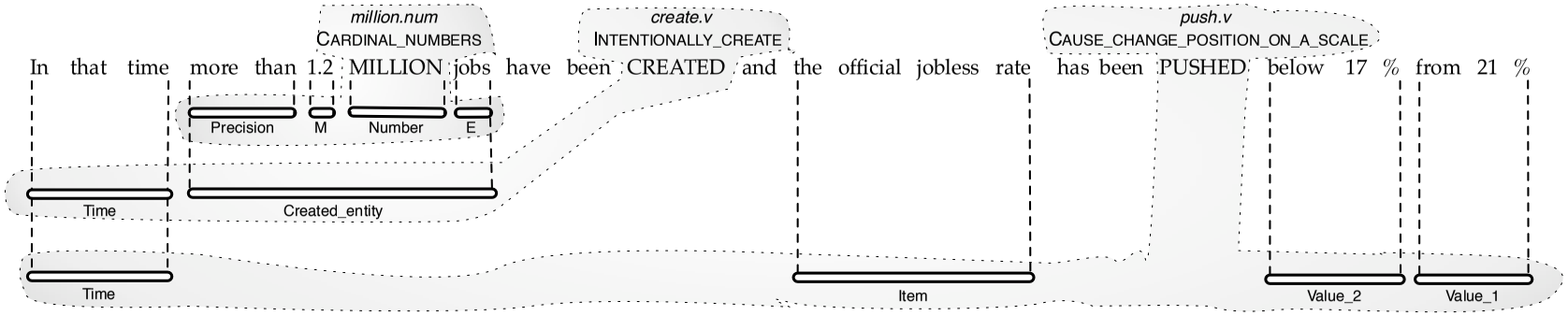

Figure 2.2: A partial depiction of frame-semantic structures for the same sentence as in Figure 2.1. The shaded shapes denote single semantic frame structures containing a semantic frame and corresponding semantic roles. The capitalized words in the sentence denote the lexical items that evoke frames. The frames and the corresponding lexical unit in the FrameNet lexicon are shown just above the frame-evoking words. For the CARDINAL_NUMBERS frame, "M" denotes the role Multiplier and "E" denotes the role Entity.

Chapter 3

Modeling Tools

Here, we describe a set of tools used across the two major problems in computational semantics that we address in this thesis proposal. These tools are general, and have been used in 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. Finally §3.5 describes a word clustering algorithm to extract groups of similar words from unlabeled corpora. This chapter can be skimmed on a casual reading.

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 (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), and 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

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

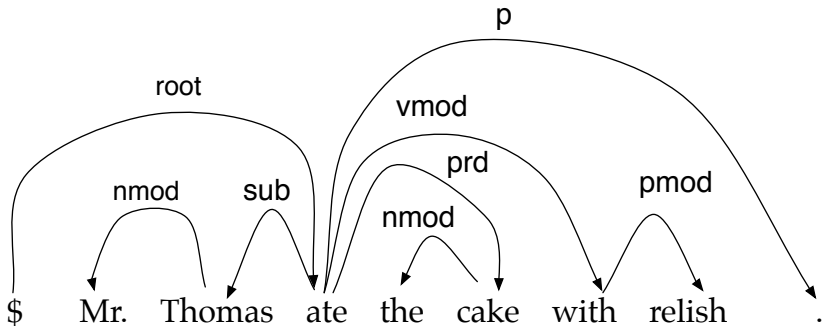

Figure 3.1: A projective dependency parse.

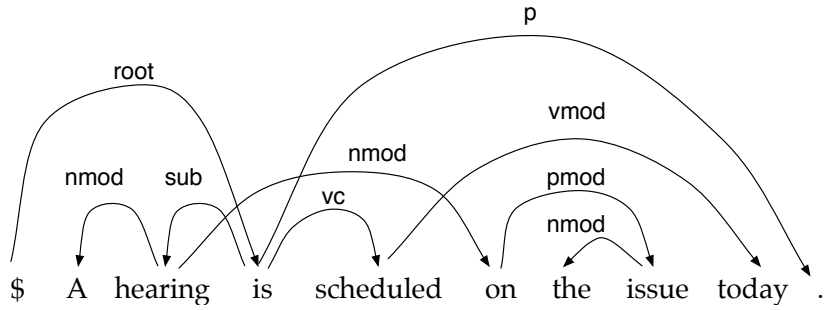

Figure 3.2: A non-projective dependency parse.

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

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:

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

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:

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

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; techniques like conditional Expectation-Maximization (Jebara and Pentland, 1999) can be used to locally optimize this function. However, in our work, we use L-BFGS to train our latent-variable models. 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

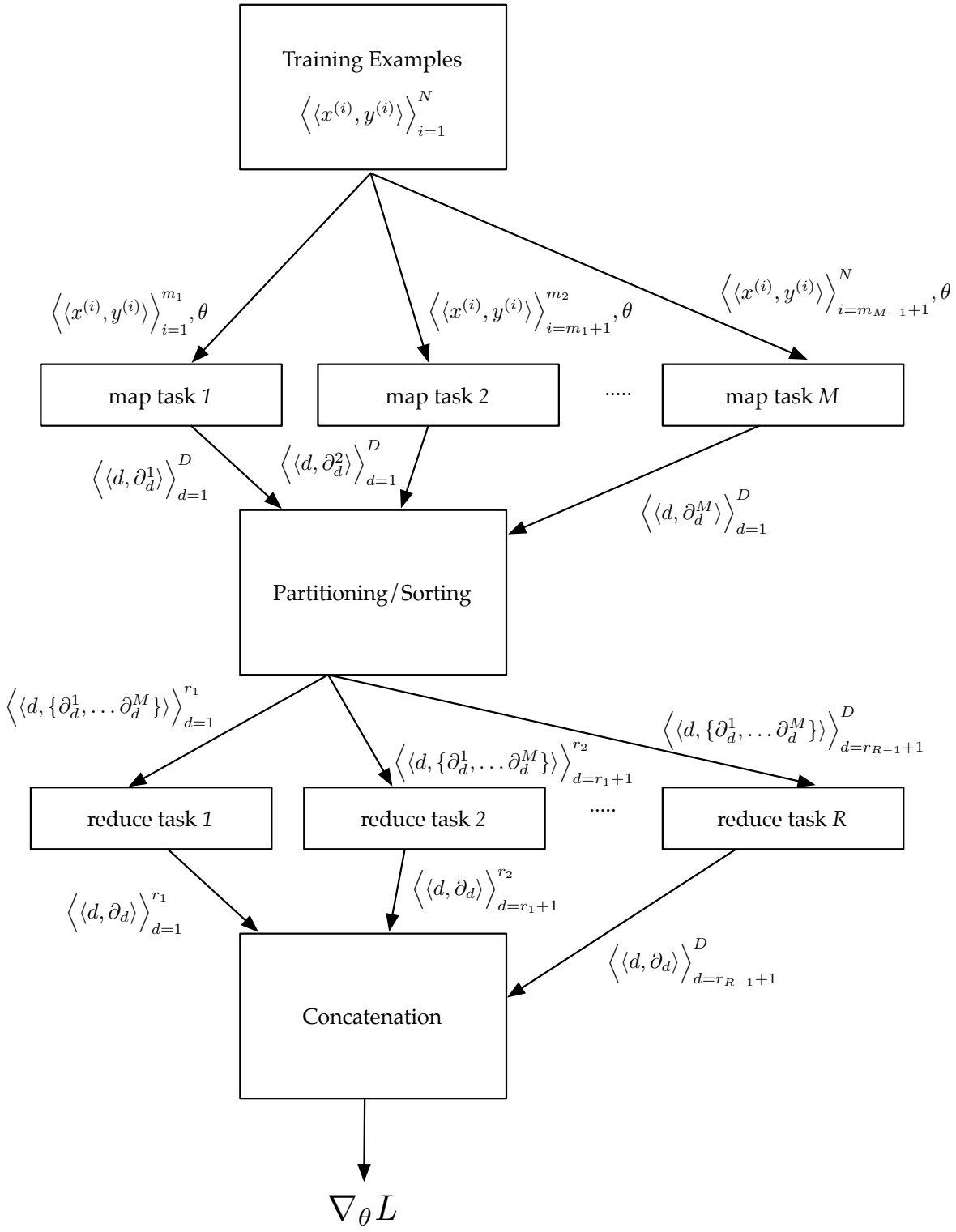

Figure 3.3: Parallelizing one iteration of gradient-based optimization.

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.

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."

3.5 Word Clusters

Deriving features from large amounts of unlabeled data can improve the performance of a supervised learning algorithm. The Brown algorithm (Brown et al., 1992) has been used to cluster words, and features derived from a word's cluster memberships have improved syntax processing tasks like dependency parsing (Koo et al., 2008). However, Koo et al. (2008) used a corpus of only 43 million tokens to derive the word clusters. Very recently, Lin and Wu (2009) used news data from the Linguistic Data Consortium containing the English Gigaword, the Tipster corpus and Reuters RCV1 corpora containing a total of 3.4 billion tokens, and a web scale corpus containing 700 billion tokens to perform word and phrase clustering using the straightforward the K-Means algorithm (MacQueen, 1967). Their clusters improved the tasks of named-entity recognition and web query classification over a vanilla supervised setting. Furthermore, Lin and Wu (2009) used MapReduce to parallelize the K-Means algorithm very efficiently, and accelerated the process of gathering clusters.

In our work, we follow Lin and Wu (2009), but limit ourselves to the English Gigaword (Graff, 2003) containing 128 million sentences and 3.02 billion tokens, and only perform word clustering. We choose only those words in our dataset that occur more than 100 times in the Gigaword, resulting in a vocabulary of 179,468 types. We depart from previous work by implementing the K-Means++ algorithm (Arthur and Vassilvitskii, 2007) instead of vanilla K-Means. The K-Means++ algorithm initializes the K cluster centroids intelligently, and has exhibited better convergence rate and better quality of clusters for several applications. The initialization algorithm is as follows:

1. Let the set of feature vectors for each type in the vocabulary be denoted as $\mathcal{V}$.

2. An initial centroid $\mathcal{C}_1$ is chosen uniformly at random from $\mathcal{V}$.

3. The next centroid $\mathcal{C}_i$ is chosen selecting $\mathcal{C}_i = v' \in \mathcal{V}$ with probability $\frac{D(v')^2}{\sum_v D(v)^2}$. Here, $D(v) = \min \mathrm{d}(v, u), u \in \{c_1, \ldots, c_{i-1}\}$ where d is a distance function.

4. Step 3 is repeated until we have chosen a total of K centroids.

After selecting the initial K centroids using the above algorithm, we implement the traditional K-Means algorithm on MapReduce as follows:

1. The feature vectors for all the words in the vocabulary are distributed among several map tasks. Each map task also receives the vectors for the K centroids.
2. Each map task computes the closest centroid for each feature vector, and outputs a key-value pair containing the centroid ID as key and the feature vector as value.
3. A reduce task receives a centroid ID as key and all its associated feature vectors as values. Next, each reduce task computes the average of all the received feature vectors, and produces the centroid ID as key and the averaged result as a vector value.
4. Steps 1-3 are repeated till convergence, starting with the new set of K centroids.
5. The word clusters are derived by grouping words on the basis of their closest centroids.

There has been considerable previous work investigating different ways of creating the feature vectors for a particular word. For phrase clustering, Lin and Wu (2009) investigated two ways of constructing the feature vectors. In the first, they chose words that appeared within a 1-word context around a given word type. All such surrounding words were gathered from the corpus along with their counts. Each such surrounding context word was treated as a feature. The feature value was computed as the pointwise mutual information (PMI) of the context word and the word to be clustered. In their second type of feature computation, they considered all words within a 3-word context window, and computed the feature vectors in a similar fashion. The former resulted in more categorical features while the latter generated topical clusters. Another avenue of computing features can be the use of syntactic contexts like head-modifier relationships, which result in clusters containing similar meaning words.

In our work till now, we have considered only the surrounding words contained in a 1-word window around a given word as features, and used PMI to compute a feature's value. We experimented with several values of K, namely 64, 128, 256, 512, 1024, and plan to use them as features in our probabilistic models for semantics. Figure 3.4 shows a few coherent clusters found using the K-Means++ algorithm with $K = 512$.

Type	Examples
Adverbs ending with *-ly*	erroneously, mechanically, customarily, inappropriately, inexorably, remorselessly, consciously, gleefully, unfairly, hastily, frenetically, loyally, callously, errantly systemically, valiantly, unofficially, ...
Belonging to a region	andorran, ossetian, malian, azeri, qatari, rican, kirgiz, uighur, fijian, isreali, jamaican, slovakian, tanzanian, kosovan, british, antiguan, then-soviet, salvadoran, canadian, kirghiz, rwandese, gibraltarian, slovene, bavarian, emirati, palestinan, spanish, croat, ...
Indian names	saroj, bikram, tirath, binod, sharda, subhas, kajal, a.v., venkat, k.l., abhijit, kalpana, pravin, rishi, k.r., surya, chetan, mohandas, shyam, suman, ...
Stores	cash-and-carry, k-mart, panera, galeries, tesco, high-street, home-furnishings, hardee, randalls, walmart, eckerd, osco, coffeeshop, carmike, office-products, ...
Digital entities	narrowband, linux, e-business, photoshop, magnifier, data-base, internet-enabled, 1080p, k56flex, multiple-access, wcdma, add/remove, always-on, piezoelectric, 56-kilobit, hypertext, off-the-shelf, ftp, dial-in, ...
n^{th}	192nd, 187th, 442nd, 196th, 171st, 205th, 13th, 18th, 124th, 97th, 6th, 15th, twelfth, 131st, 320th, 123rd, 24th, 12th, tenth, 16th, 60th, 59th, 85th, 256th, 172nd, 198th, 203rd, eighteenth, 138th, 111th, nineteenth, ...
music related	cumbia, musical-theater, indie, ranchera, coloratura, hillbilly, adult-contemporary, zydeco, early-music, acoustic, go-go, minstrel, hip-hop, rock-and-roll, top-40, narco, soft-rock, ...
words ending with *-ing*	vaccinating, air-dropping, dooming, petitioning, disobeying, commercializing, fortifying, airlifting, emigrating, recasting, simulating, supplementing, expediting, burnishing, redrawing, redressing, rationalizing, short-circuiting, skewing, ...

Figure 3.4: Example word clusters obtained using the K-Means++ algorithm. The types of the clusters are manual labels.

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 original 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 $\mathbf{s}$ should be "inspired" by the structure of $\mathbf{s}$.

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).

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 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 $\mathbf{s}$," when $c = \mathsf{p}$.) Given $\mathbf{s}$, G_c generates the other sentence in the pair, $\mathbf{s}'$.

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 $\mathbf{s}$ and which was $\mathbf{s}'$). Both orderings are assumed to be equally probable. For class c,

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

where c can be p or n; $G_{\mathsf{p}}(\mathbf{s})$ is the QG that generates paraphrases for sentence $\mathbf{s}$, while $G_{\mathsf{n}}(\mathbf{s})$ is the QG that generates sentences that are *not* paraphrases of sentence $\mathbf{s}$. This latter model may seem counter-intuitive: since the vast majority of possible sentences are not paraphrases of $\mathbf{s}$, 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.

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

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), 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 sentence. 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 how we model $p_Q(\mathbf{t} \mid G_{\mathsf{p}}(\mathbf{s}))$ 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

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

[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.

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

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

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, $t_{\tau^{\mathbf{t}}_p(i)}$, is aligned to s_l. For leaves of $\tau^{\mathbf{t}}$, the base case is:

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

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

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

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 $l = a(j)$.

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

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 $\tau^{\mathbf{t}}$, 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]

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

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 selected.

Dependency label, POS, and named entity class The newly generated target word's dependency label, POS, and named entity class 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 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

[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.

Configuration	Description
parent-child	$\tau_p^{\mathbf{s}}(a(i)) = a(j)$, appended with $\tau_l^{\mathbf{s}}(a(i))$
child-parent	$a(i) = \tau_p^{\mathbf{s}}(a(j))$, appended with $\tau_l^{\mathbf{s}}(a(j))$
grandparent-grandchild	$\tau_p^{\mathbf{s}}(\tau_p^{\mathbf{s}}(a(i))) = a(j)$, appended with $\tau_l^{\mathbf{s}}(a(i))$
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 $\mathbf{s}$
child-null	$a(i) = 0$
parent-null	$a(j) = 0$, $a(i)$ is something other than root of $\mathbf{s}$
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.

$s_{a(i)}$.

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:

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

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).

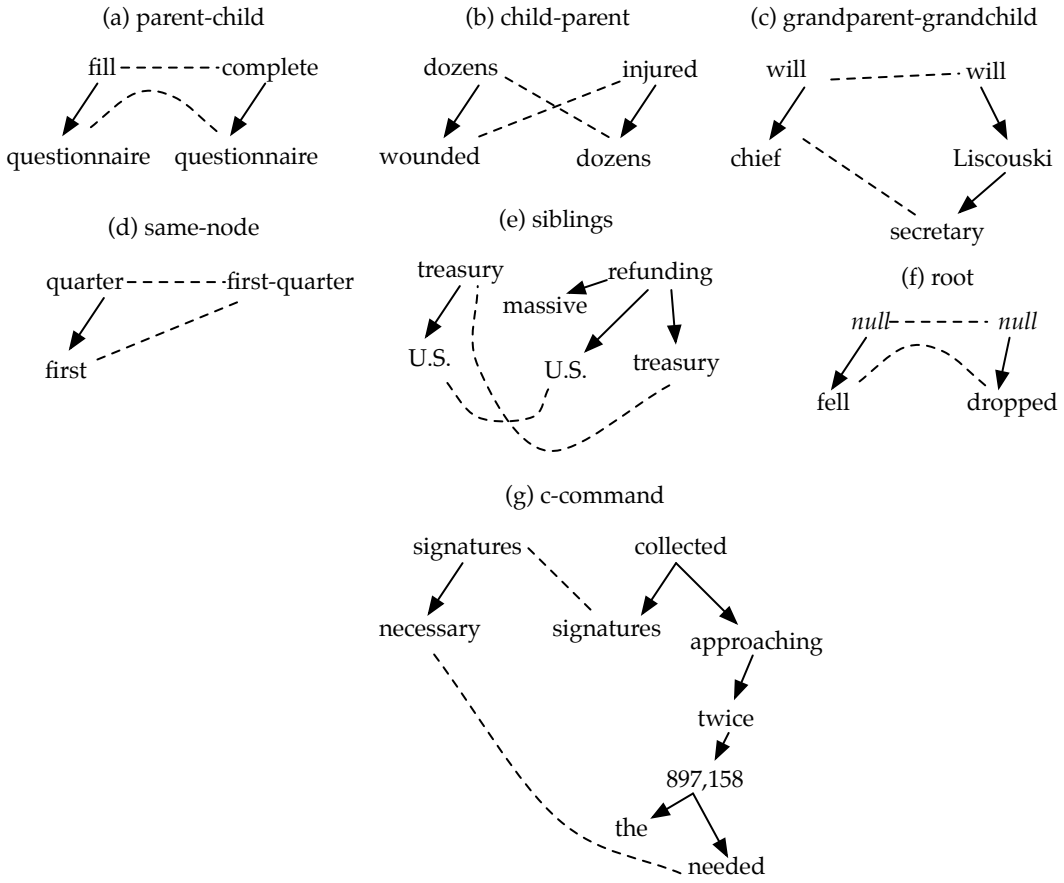

Figure 4.1: Some example configurations from Table 4.1 that G_{p} discovers in the dev. data. Directed arrows show head-modifier relationships, while dotted arrows show alignments.

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 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 sentences 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:

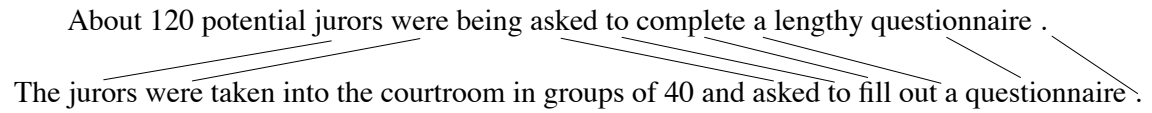

Figure 4.2: Discovered alignment of Example 4.18 produced by G_{p}. Observe that the model aligns identical words and also "complete" and "fill" in this specific case. This kind of alignment provides an edge over a simple lexical overlap model.

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

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 probabilities 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.

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 4.19 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, making a decision difficult for a naïve lexical overlap classifier. (In fact, 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.

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.

	Model	**Accuracy**	**Precision**	**Recall**
	all p	66.49	66.49	100.00
baselines	Wan et al. SVM (reported)	75.63	77.00	90.00
	Wan et al. SVM (replication)	75.42	76.88	90.14
	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*

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 $p_Q(\mathsf{p} \mid \mathbf{s}_1, \mathbf{s}_2)$ is greater than 0.5, the pair is labeled "paraphrase" (as in Equation 4.1).

4.4.1 Baseline

We replicated a state-of-the-art baseline model for comparison. Wan et al. (2006) report the best published accuracy, to our knowledge, on this task, using a support vector machine. 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).[6] 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).

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,[7] 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.

[6] `http://svmlight.joachims.org`

[7] 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.

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. 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):

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

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, achieving an accuracy of 76.06%.[8] 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

In this chapter, 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 have shown that this model is competitive for determining whether there exists a semantic relationship between them, and can be improved by principled combination with more standard lexical overlap approaches.

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.

Our method is amenable to the use of more features at the alignment sites because it is composed of locally normalized log-linear models; it can be extended to capture properties like distributional similarity, and lexical features derived from unlabeled data can be incorporated with ease (see §6.2.2).

[8] 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

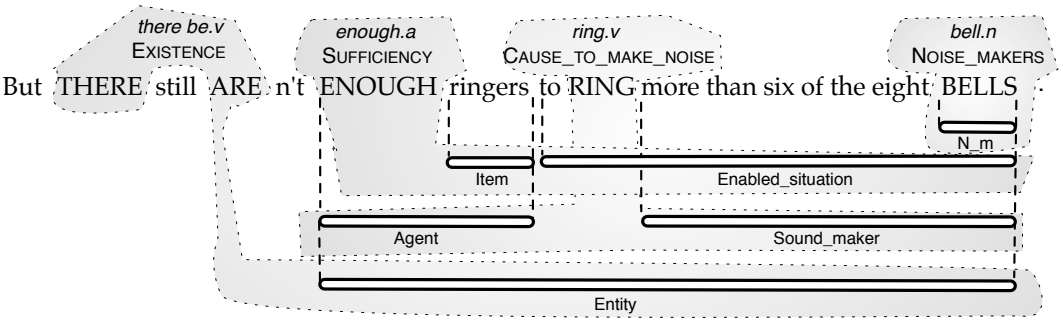

Figure 5.1: A sentence from PropBank and the SemEval'07 training data, and a partial depiction of gold FrameNet annotations. Each frame appears above the target that evokes it, and each target is capitalized. Lexical units are shown above the frames. Each shaded figure denotes a frame and all its roles. The double stroked lines denote roles. "N_m" under *bells* is short for the Noise_maker role of the NOISE_MAKERS frame—it is a **denoted frame element** because it is also the target. The last row indicates that *there...are* is a discontinuous target. In PropBank, the verb *ring* is the only annotated predicate for this sentence, and it is not related to other predicates with similar meanings.

FrameNet (Fillmore et al., 2003) is a rich linguistic resource containing 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 semisupervised learning. 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, and a precision-boosting constraint that forbids

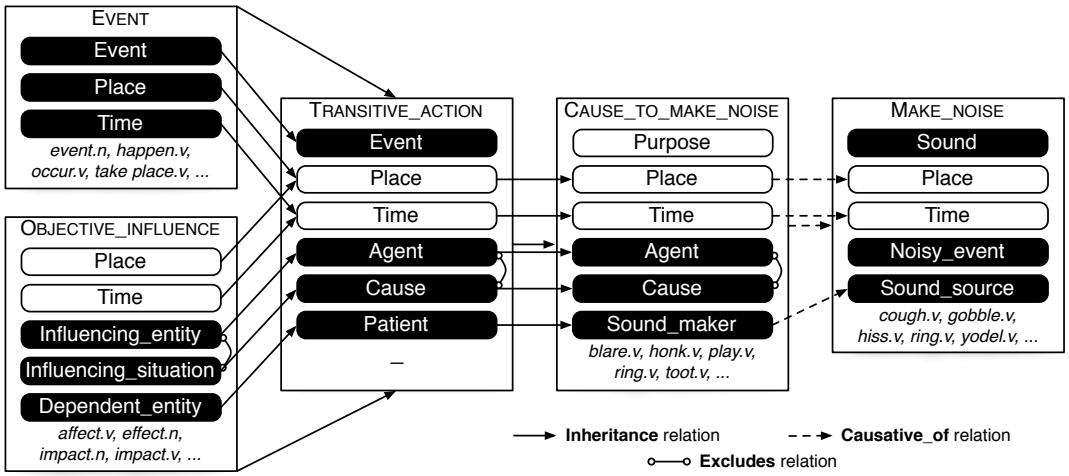

Figure 5.2: Partial illustration of frames, roles, and LUs related to the CAUSE_TO_MAKE_NOISE frame, from the FrameNet lexicon. "Core" roles are filled ovals. Non-core roles (such as Place and Time) as unfilled ovals. No particular significance is ascribed to the ordering of a frame's roles in its lexicon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

arguments of the same predicate to overlap. 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 in (Das et al., 2010a) and (Das et al., 2010b).

This chapter is organized as follows. §5.1 describes in detail the task and the resources we used to solve it. §5.2, §5.3 and §5.4 detail the three major subtasks that constitute the full problem of frame-semantic parsing, and also present the experiments and the results achieved. Finally, §5.5 concludes this chapter with a discussion.

5.1 Resources and Task

We consider frame-semantic parsing resources.

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 several lemmas (with part 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

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

[2]Like the SemEval'07 participants, we used FrameNet v. 1.3 (`http://framenet.icsi.berkeley.edu`).

FRAMENET LEXICON V. 1.3		
lexical entries	**exemplars**	
	counts	*coverage*
8379 LUs	139K sentences, 3.1M words	70% LUs
795 frames	1 frame annotation / sentence	63% frames
7124 roles	285K overt arguments	56% roles

Table 5.1: Snapshot of lexicon entries and exemplar sentences. Coverage indicates the fraction of types attested in at least one exemplar. The lexicon associates an average of 12.8 LUs with a frame, and 66% of those LUs are attested for that frame. The average ambiguity of an LU is 1.2 frames (the 1322 ambiguous LUs have an average ambiguity of 2.4 frames).

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.

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. We found that using exemplar sentences directly to train our models hurt performance as evaluated on SemEval'07 data, even though the number of exemplar sentences is an order of magnitude larger than the number of sentences in our training set (§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).

TARGETS AND ARGUMENTS BY PART OF SPEECH					
	targets			**arguments**	
	count	%		*count*	%
Noun	5155	52	Noun	9439	55
Verb	2785	28	Preposition or		
Adjective	1411	14	complementizer	2553	15
Preposition	296	3	Adjective	1744	10
Adverb	103	1	Verb	1156	7
Number	63	1	Pronoun	736	4
Conjunction	8		Adverb	373	2
Article	3		Other	1047	6
	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.

FULL-TEXT ANNOTATIONS	*SemEval'07 data*								
	train			**dev**			**test**		
Size	*(words sentences documents)*								
all	43.3K	1.7K	22	6.3K	251	4	2.8K	120	3
ANC (travel)	3.9K	154	2	.8K	32	1	1.3K	67	1
NTI (bureaucratic)	32.2K	1.2K	15	5.5K	219	3	1.5K	53	2
PropBank (news)	7.3K	325	5	0	0	0	0	0	0
Annotations	*(frames/word overt arguments/word)*								
all	0.23	0.39		0.22	0.37		0.37	0.65	
ANC	0.22	0.38		0.15	0.29		0.37	0.60	
NTI	0.23	0.40		0.23	0.37		0.38	0.69	
PropBank	0.22	0.37							
Coverage of lexicon	*(%_frames %_roles %_LUs)*								
all	64.1	27.4	21.0	34.0	10.2	7.3	29.3	7.7	4.9
ANC	26.4	7.4	4.8	8.9	2.0	1.1	17.5	3.9	2.3
NTI	52.4	21.1	14.9	31.5	9.2	6.7	19.0	5.0	3.0
PropBank	40.8	12.0	8.4						
Out-of-lexicon types	*(frames roles LUs)*								
all	14	69	71	2	4	2	39	99	189
ANC	12	39	41	0	0	2	26	63	123
NTI	6	32	33	2	4	0	19	45	70
PropBank	3	11	3						
Out-of-lexicon tokens	*(%_frames %_roles %_LUs)*								
all	0.7	0.9	1.1	1.0	0.4	0.2	9.8	11.2	25.3
ANC	3.2	4.2	7.6	0.0	0.0	1.8	11.5	13.5	34.8
NTI	0.6	0.6	0.5	1.1	0.4	0.0	8.5	9.4	17.4
PropBank	0.3	0.4	0.2						

Table 5.3: Snapshot of the SemEval'07 annotated data. Our development set encompasses the following documents: `StephanopoulousCrimes` (from ANC), plus `Iran_Biological`, `NorthKorea_Introduction`, and `WMDNews_042106` (NTI). We use the standard test set, consisting of `IntroOfDublin` (ANC) and `chinaOverview` and `workAdvances` (NTI). Two ANC documents provided as part of the task were unannotated; we ignore them throughout.

5.1.2 Data

Our training, development, and test sets consist of documents annotated with frame-semantic structures for the SemEval'07 task, which we refer to collectively as the **SemEval'07 data**.[3] For the most part, the frames and roles used in annotating these documents were defined in the FrameNet lexicon, but there are some exceptions for which the annotators defined supplementary frames and roles; these are included in the possible output of our parser.

Table 5.3 provides a snapshot of the SemEval'07 data. We randomly selected three documents from the original SemEval training data to create a development set for tuning model hyperparameters. Notice that the test set contains more annotations per word, both in terms of frames and arguments. Moreover, there are many more out-of-lexicon frame, role, and LU types in the test set than in the training set. This inconsistency in the data results in poor

[3]The full-text annotations and other resources for the 2007 task are available at `http://framenet.icsi.berkeley.edu/semeval/FSSE.html`.

recall scores for all models trained on the given data split, a problem we have not sought to address here.

Table 5.2 shows the breakdown of the targets and the arguments with respect to part of speech in the SemEval'07 training data. The statistics indicate that for both, nouns dominate the annotations, followed by verbs. However, unlike other corpora for semantic role labeling the FrameNet annotations encompass nearly all types of POS for the targets.

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 (Baker et al., 2007). For our experiments, statistical significance is measured using a reimplementation of Dan Bikel's randomized parsing evaluation comparator.[4]

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.

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

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 5.4: Target identification results for our system and the baseline. 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 predicates from non-predicates. 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, syntactially-dependent noun (*request*). Furthermore, only temporal, locative, and directional senses of prepositions evoke frames.

We found that, because the test set is more completely annotated—that is, it boasts far more frames per token than the training data (see Table 5.3)—learned models did not generalize well and achieved poor test recall. 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 the SemEval'07 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.[5]

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, 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.

Results. Table 5.4 shows results on target identification; 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 tokens (e.g. *chemical and biological weapon*.N,

[5]There are 629 multiword LUs in the lexicon, and they correspond to 4.8% of the targets in the 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).

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.

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 (see Table 5.1).[6] All targets in the exemplar sentences, and most in our training and test data, correspond to known LUs (see Table 5.3).

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 in our training set. Let $\mathcal{L}_f \subseteq \mathcal{L}$ be the subset of these targets annotated as evoking a particular frame f.[7] 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 SemEval annotators created 47 new frames not present in the lexicon, out of which 14 belonged to our training set. We considered these with the 795 frames in the lexicon when parsing new data. 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).

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).[8] Let t^l_i 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

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

[7] On average, there are 34 targets per frame in our dataset. The average frame ambiguity of each target in $\mathcal{L}$ is 1.17.

[8] Each t_i is a word sequence $\langle w_u, \ldots, w_v \rangle$, $1 \leq u \leq v \leq n$, though in principle targets can be noncontiguous

$t_i^l \in \mathcal{L}_f^l$—or if $t_i^l \notin \mathcal{L}^l$, then every known frame (the latter condition applies for 4.7% of the gold targets in the development set). In both cases, we let $\mathcal{F}_i$ be the set of candidate frames for the ith target in $\mathbf{x}$.

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:

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

We adopt a conditional log-linear model: for $f \in \mathcal{F}_i$ and $\ell \in \mathcal{L}_f$,

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

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. 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 instantiates 662,020 binary features, which are detailed in Table 5.5.

5.3.3 Training

Given the training subset of the SemEval'07 data, which is of the form $\left\langle \langle \mathbf{x}^{(j)}, \mathbf{t}^{(j)}, \mathbf{f}^{(j)}, \mathcal{A}^{(j)} \rangle \right\rangle_{j=1}^{N}$ ($N = 1663$ is the number of sentences), we discriminatively

[9] 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 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.

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

[11] POS tags are found automatically during preprocessing.

• the POS of the parent of the head word of t_i
• the set of syntactic dependencies of the head word[9] 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[10] ρ holds between ℓ and t_i
• WordNet relation[10] ρ holds between ℓ and t_i, and the prototype is ℓ
• WordNet relation[10] 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 5.5: 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[11] of a target seen in an exemplar or training sentence as evoking f. There are a total of 662,020 binary features in our model.

train the frame identification model by maximizing the following log-likelihood:[12]

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

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. This is the most expensive model that we train: with 100 CPUs parallelized using MapReduce (see §3.3), training takes several hours. (Decoding takes only a few minutes on one CPU for the test set.)

5.3.4 Results

We evaluate the performance of our frame identification model given gold-standard targets and automatically identified targets (§5.2); see Table 5.6.

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!5.6 (though this is not significant).

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

FRAME IDENTIFICATION (§5.3)	*targets*	exact frame matching *P*	*R*	F_1	partial frame matching *P*	*R*	F_1
Frame identification (oracle targets)	$*$	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*

Table 5.6: Frame identification results. 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 ($p < 0.05$).

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.

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.[13]

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 identified by J&N'07 and frames classified by our frame identification model resulted in scores at 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 leads to 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.

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

[13] 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.

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 81% of arguments in the development data. The empty span, denoted $\emptyset$, is also included in $\mathcal{S}$, since some roles are not explicitly filled; in the 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.[14] In training, if a labeled argument is not a valid subtree of the dependency parse, we add its span to $\mathcal{S}$.

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:

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

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 training data.

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).

[14] 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 instantiations.

Our role-filling model, unlike a sentence-global argument detection-and-classification approach,[15] permits this sort of argument sharing among frames. The incidence of span overlap among frames is much higher; Figure 5.1 illustrates a case with a high degree of overlap. 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 $s \in \mathcal{S}$.[16] For features using the head word of the target t or a candidate argument span s, we use the heuristic described in footnote 9 for selecting the head of non-subtree spans. Table 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.[17] 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 a detailed analysis of the effects of using role features at varying levels of granularity.

5.4.2 Training

We train the argument identification model by:

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

This objective function is concave, and we globally optimize it using stochastic gradient ascent (Bottou, 2003). We train this model until the argument identification F_1 score stops increasing on the development data. Best results on this dataset were obtained with a batch size of 2 and 23 passes through the data.

5.4.3 Approximate Joint Decoding

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 single frame cannot be filled by overlapping spans.[20] 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

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

[16] 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.

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

[18] Quantized into groups: $(-\infty, -20]$, $[-19, -10]$, $[-9, -5]$, -4, -3, -2, -1, 0, 1, 2, 3, 4, $[5, 9]$, $[10, 19]$, $[20, \infty)$.

[19] We treat as a closed-class POS tag any Penn Treebank tag except for CD which does not start with V, N, A, or R.

[20] 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.

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	● $\lvert s \rvert$, the number of words in the candidate argument[18]
○ the first word of s has POS π, provided $\lvert s \rvert > 0$	○ the head word of s has syntactic dependency type τ
○ the last word of s has POS π, provided $\lvert s \rvert > 0$	● 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}, provided that π_{s_1} is a closed-class POS[19]	● τ_{s_2}, provided that $\lvert s \rvert \geq 2$
● w_{s_2} and its closed-class POS tag π_{s_2}, provided that $\lvert s \rvert \geq 2$	● $\tau_{s_{\lvert s \rvert}}$, provided that $\lvert s \rvert \geq 3$
● the last word of s: $w_{s_{\lvert s \rvert}}$, and its closed-class POS tag $\pi_{s_{\lvert s \rvert}}$, provided that $\lvert s \rvert \geq 3$	○ the first word of s has lemma λ, provided $\lvert s \rvert > 0$
	○ the head word of s has lemma λ
	○ the last word of s has lemma λ, provided $\lvert s \rvert > 0$
◐ 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)	
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[18]
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[18]	○ linear word distance between the middle word of s and the middle word of t, provided s and t do not overlap[18]

Table 5.7: Features used for argument identification. Instantiating the above (binary) features for our data yields 1,297,857 parameters.

adopt a dynamic programming approach, which does not suit our model because we find 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

Algorithm 1 Joint decoding of frame f_i's arguments. $\mathsf{top}_\mathsf{k}(\mathcal{S}, p_{\boldsymbol{\psi}}, r_j)$ extracts the k most probable spans from $\mathcal{S}$, under $p_{\boldsymbol{\psi}}$, 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 $p_{\boldsymbol{\psi}}$ 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

Calculate $\mathcal{A}_i$ according to Equation 5.4
$\forall r \in \mathcal{R}_{f_i}$ such that $\mathcal{A}_i(r) = \emptyset$, let $\hat{\mathcal{A}}_i(r) \leftarrow \emptyset$
$\mathcal{R}^+_{f_i} \leftarrow \{r : r \in \mathcal{R}_{f_i}, \mathcal{A}_i(r) \neq \emptyset\}$
$\mathsf{n} \leftarrow |\mathcal{R}^+_{f_i}|$
Arbitrarily order $\mathcal{R}^+_{f_i}$ as $\{r_1, r_2, \ldots r_\mathsf{n}\}$
Let $D^{0:j} = \langle D_1^{0:j}, \ldots, D_\mathsf{k}^{0:j} \rangle$ refer to the k-best list of vectors of compatible filler spans for roles r_1 through r_j
Initialize $D^{0:0}$ to be empty
for $j = 1$ to n **do**
    $D^{0:j} \leftarrow \mathsf{extend}(D^{0:(j-1)}, \mathsf{top}_\mathsf{k}(\mathcal{S}, p_{\boldsymbol{\psi}}, r_j))$
end for
$\forall j \in \{1, \ldots, \mathsf{n}\}, \hat{\mathcal{A}}_i(r_j) \leftarrow D_1^{0:\mathsf{n}}[j]$
return $\hat{\mathcal{A}}_i$

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 $\mathsf{k} = 10000$.

5.4.4 Results

Performance of the argument identification model is presented in Table 5.8. The table shows how performance varies given different types of perfect input: correct targets, correct frames, and the set of correct spans; correct targets and frames, with the heuristically-constructed set of candidate spans; correct targets only, with model frames; and ultimately, no oracle input (the full frame parsing scenario).

The first four rows of results isolate the argument identification task from the frame identification task. Given gold targets and frames and an oracle set of argument spans, our local model achieves about 87% precision and 75% recall. Beam search decoding to eliminate illegal argument assignments within a frame (§5.4.3) further improves precision by about 1.6%, with negligible harm to recall. Note that 96.5% recall is possible under the constraint that roles are not multiply-filled (§5.4.1); there is thus considerable room for improvement with this constraint in place. Joint prediction of each frame's arguments is worth exploring to capture correlations not encoded in our local models or joint decoding scheme.

The 15-point drop in recall when the heuristically-built candidate argument set replaces the set of true argument spans is unsurprising: an estimated 19% of correct arguments are excluded because they are neither single words nor complete subtrees (see §5.4.1).[21] Qual-

[21] Using all constituents from the 10-best syntactic parses would improve oracle recall of spans in the develop-

ARGUMENT IDENTIFICATION					exact frame matching			partial frame matching		
	targets	*frames*	*spans*	*decoding*	*P*	*R*	F_1	*P*	*R*	F_1
Argument identification (oracle spans)	$*$	$*$	$*$	naïve	86.61	75.11	80.45			
	$*$	$*$	$*$	beam	88.29	74.77	80.97			
Argument identification (full)	$*$	$*$	model	naïve	77.43	60.76	68.09			
	$*$	$*$	model	beam	78.71	60.57	68.46			
Parsing (oracle targets)	$*$	model	model	beam	49.68	42.82	46.00	57.85	49.86	53.56
Parsing (full)	auto	model	model	beam	**58.08**	**38.76**	**46.49**	**62.76**	**41.89**	**50.24**
Parsing (J&N'07 targets and frames)	auto	model	model	beam	56.26	36.63	44.37	60.98	39.70	48.09
Baseline: J&N'07	*auto*	*model*	*model*	*N/A*	*51.59*	*35.44*	*42.01*	*56.01*	*38.48*	*45.62*

Table 5.8: Argument identification results. $*$ indicates that gold-standard labels were used for a given pipeline stage. "model" indicates that a statistical model has been used for a particular subtask. 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 ($p < 0.05$).

itatively, the problem of candidate span recall seems to be largely due to syntactic parse errors.[22] Still, the 10-point decrease in precision when using the syntactic parse to determine candidate spans 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 in this regard.

The fifth and sixth 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 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 7th 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 (last 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 powerful than the previous state of the art.

ment set by just a couple of percentage points, at the computational cost of a larger pool of candidate arguments per role.

[22]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.

5.5 Discussion

We have provided a supervised model for rich frame-semantic parsing, based on a combination of knowledge from FrameNet, two probabilistic models trained on SemEval'07 data, 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 probabilistic model for argument identification 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, and is amenable to future extension, that we consider in Chapter 6. The parser described in this chapter is available for download at `http://www.ark.cs.cmu.edu/SEMAFOR`.

Chapter 6

Proposed Work

This chapter proposes extensions to the models described in Chapters 4 for modeling paraphrase and in Chapter 5 for frame-semantic parsing, as future work to be completed for this dissertation. The common aspect across the different proposed directions is the use of large amounts of unlabeled data to aid in better learning of natural language semantics. The choice of using unlabeled data to append purely supervised methods follows from the absence of large volumes of training data for the problems we consider in this thesis, and we aim to explore the possibilities of improving the quality of the semantic processing tasks using raw text.

6.1 Background and Motivation

The use of unlabeled data along with annotated training examples has been widely researched in the NLP community. First, unlabeled data has been used in scenarios like domain adaptation where labeled data in varied domains is unavailable in large quantities and models from one domain is projected to another. Second, unlabeled data has been successfully used along with labeled data for scenarios where labeled data for a single domain is very limited, thus making it hard for a trained model to generalize to unseen data. For the semantic processing tasks we consider in this thesis, we face the latter scenario. This is especially true for the task of frame-semantic parsing, where the training data size is miniscule, containing less than 2,000 sentences, in contrast to tasks like syntactic parsing, where the most popular dataset in English contains around 40,000 example sentences. Hence, we propose to use techniques that derive useful information from unlabeled data with the hope of improving upon supervised models described in the previous sections.

There have been attempts at incorporating features based on word clusters derived from unlabeled data in supervised discriminative models. Miller et al. (2004) presented a technique of obtaining hierarchical word clusters from raw text and used the cluster information as features in named-entity recognition. This approach has been used later for dependency parsing by Koo et al. (2008) and for query classification by Lin and Wu (2009). The latter went further and clustered phrases using a web-scale text corpus and used the cluster information for discriminative learning. There are a few instances of research in semantic processing of language that used word clusters derived from raw text. He and Gildea (2006) derived clusters using distributional similarity (Pereira et al., 1993) and used the clusters to improve semantic role labeling using a probabilistic model. However, the scale at which He and Gildea (2006) performed their experiments was much smaller than the task of frame-

semantic parsing that we consider in our work. More recently, Deschacht and Moens (2009) have used a latent-variable model that finds similarity between words using unlabeled text, and used it to improve PropBank style SRL performance. In our work, we plan to leverage word clusters of various types gathered from the Gigaword corpus (Graff, 2003) for the tasks of paraphrase identification and frame-semantic structure extraction. Gigaword-scale data for word clustering has been used only by Lin and Wu (2009) and they showed significant improvements for shallow NLP tasks. We hope to attain lexical semantic knowledge from the word clusters to improve semantic analysis.

Another major area of research is the development of statistical models that learn from both labeled and unlabeled data. This is traditionally named *semi-supervised learning*, and a large research community has focused on these techniques and have applied them for language processing tasks. Here, we will cite a few influential papers that used semi-supervised learning successfully for NLP. Steedman et al. (2003) used *co-training* (Blum and Mitchell, 1998) for bootstrapping syntactic parsers using labeled and unlabeled data, and received improvements over vanilla supervised methods. Since co-training requires statistical classifiers using diverse views of a particular problem, they used two parsers, one based on probabilistic context free grammar and the other on lexicalized Tree Adjoining Grammar (Joshi and Schabes, 1997). *Self-training* is another approach that starts by learning from labeled data and then labels unlabeled data with the trained model. Some automatically labeled samples (usually on which the model has high confidence) from the originally unlabeled data are selected and appended to the training set, and then the model is retrained. This is done iteratively till some stopping criterion. McClosky et al. (2006) used this approach for syntactic parsing to achieve better performance than supervised parsers. More recently, Chang et al. (2007) presented a method of using constraints (encoding domain knowledge) in semi-supervised learning. Their method started by learning a model from labeled data. Next they used this learned model to label unlabeled data along with a set of manually set constraints, and chose the top K labels for each unlabeled data point. All the data points in the unlabeled set with their top K labels are next used to learn a model. The model initially learned from the supervised data and the model learned using the unlabeled data are interpolated to result in a new model. This new model is used to label the unlabeled data again and a newer model is learned. This process is repeated several times. This algorithm described by Chang et al. (2007) resulted in improved performance compared to supervised methods.

Recently, NLP researchers using probabilistic methods for various problems, have attempted to estimate statistical models from both labeled and unlabeled data *together* by incorporating a component that minimizes a function modeling the unlabeled data, appending that to the standard MAP estimation framework for supervised learning (equations 3.3-3.4). One example of such a framework is *expectation regularization* for semi-supervised learning (Mann and McCallum, 2007) where the objective function of conditional log-linear models is appended by a divergence term between the expected value of feature/label pairs predicted by the model and human provided feature/label expectation priors. This method needs human provided class priors given the features in the model, or these priors can be estimated from labeled data, which is very small for our case. Another similar technique is the use of entropy minimization where the objective function of supervised log-linear models is appended with the negative conditional entropy of the unlabeled data (Grandvalet and Bengio, 2004; Jiao et al., 2006). Significant gains in information extraction tasks and syntactic parsing have been observed using this technique. We propose to adopt this framework for our models of semantics. Extending our log-linear models with the entropy

of unlabeled data is straightforward and features gathered from unlabeled data using this technique shows definite hope of improving over supervised frame-semantic parsing.

Another attractive avenue of research is the adoption of a hybrid framework of discriminative models and generative models. The general idea behind such a framework is the assumption that the unlabeled data is explained by a generative process, while the supervised data labels are explained by a discriminative model given sets of features who weights are estimated both from labeled as well as unlabeled data (Suzuki and Isozaki, 2008). This approach is attractive in that it combines conditional models with EM-like generative models that are used for unsupervised learning. It also does not involve heavy computation for calculating complex derivative terms unlike other semi-supervised learning methods, making the method amenable to the use of huge amounts of unlabeled data. We propose to use this approach for the semantic processing tasks at hand.

Our proposed directions of future work can be summarized into three distinct types and are inspired by relevant work that exploits both labeled and unlabeled data:

1. We plan to incorporate categorical, topical, and synonymous **word clusters** (see description of clustering methods in §3.5) into our models for paraphrase identification and frame-semantic parsing, with the goal of injecting information extracted from unlabeled corpora, and hope to improve upon purely supervised models for both the tasks. §6.2 focuses on this particular direction of future work.

2. The argument identification phase of frame-semantic parsing deals with 7124 unique roles that are listed in the FrameNet lexicon. The total number of overt roles that serve as training instances for the log-linear model devised for the argument identification task (see §5.4) is only around three times the total number of role types in the lexicon. This suggests that the number of argument phrases in the training data per role type is very few, making the features used for the model sparse. We intend to use **phrasal paraphrases** and **phrase clusters** gathered from large volumes of unlabeled corpora as features in our argument identification model to improve our model's performance, and we focus on this aspect in §6.3.

3. Our final goal as a part of this thesis is to investigate **semi-supervised learning** techniques to improve the task of frame-semantic parsing. To this end, we plan to investigate entropy minimization techniques first. We also intend to devise techniques that combine discriminative and generative models into one framework and leverage a lot of unlabeled data to improve upon supervised learning. §6.4 focuses on this part of proposed work.

6.2 Word Clusters and Semantics

Recent work in syntactic analysis (Koo et al., 2008) and information extraction (Lin and Wu, 2009) has showed that using word cluster information in discriminative models can significantly improve model performance over vanilla supervised methods. In this section, we concentrate on incorporating word cluster information for two semantic processing tasks – paraphrase identification and frame-semantic parsing.

In §3.5, we described a general procedure of deriving word clusters from unlabeled corpora, and enumerated a set of coherent clusters extracted from the Gigaword corpus using K-Means++ clustering. Till date, we have been able to run K-Means++ clustering using

features derived from minimal contextual information of only 1-word windows around a word to be clustered. This resulted in *categorical* clusters, as shown in Figure 3.4. Lin and Wu (2009) found that using larger context windows, namely that of 3-word windows around a word resulted in more *topical* clusters, at least for the case of phrase clustering, which they investigated.

As proposed work, we plan to run our clustering algorithms for larger context windows of 3 words around a given word to be clustered, and analyze the clusters qualitatively. If qualitative differences are found when compared with the clusters derived using 1-word context, we plan to use these clusters for further experiments. At this stage, feature extraction using 3-word context for every word in the Gigaword vocabulary is complete, along with the special initialization step for K-Means++; only the clustering iterations for various number of centroids K is remaining. We plan to use MapReduce to parallelize the clustering algorithm, as described in §3.5.

Furthermore, we plan to run a final set of clustering experiments, where we parse the entire Gigaword corpus using the MST parser (see §3.1), and extract features based on the syntactic context of a word to be clustered. We define the syntactic context of a word w as a tuple (r, w'), where r is a syntactic relation in the dependency tree and w' is the word to which w is connected to. These syntactic contexts serve as features instead of just the words, as in the clustering method described in §3.5; such syntactic context features have been used in a wide body of work in finding distributional similarity between words (Lin and Pantel, 2001; Gorman and Curran, 2006). Previous work shows that clusters derived by using syntactic contexts are groups of similar meaning or *synonymous* words. We intend to use these clusters in our experiments too.

6.2.1 Word Clusters in Frame-Semantic Parsing

Word cluster information can be appended as features to the two major components of frame-semantic parsing, namely frame identification §5.3 and argument identification §5.4. We call this set of proposed experiments WC1. A possible set of features that could be appended to the existing feature set shown in Table 5.5 is tabulated in in Table 6.1.

• the *categorical* word cluster of the parent of the head word of t_i
• the *topical* word cluster of the parent of the head word of t_i
• the lemmatized sequence of words in the prototype and their *categorical* word clusters
• the lemmatized sequence of words in the prototype and their *topical* word clusters
• the sequence of *synonymous* word clusters corresponding to the words in t_i
• the sequence of *synonymous* word clusters corresponding to the words in t_i and the sequence of *synonymous* word clusters corresponding to the words in $\mathbf{w}_\ell$
• a binary feature indicating whether the sequence of *synonymous* word clusters in t_i and the sequence of *synonymous* word clusters in $\mathbf{w}_\ell$ are the same

Table 6.1: Proposed features to be used for frame identification. All features incorporate f, the frame being scored. $\mathbf{w}_\ell$ represents the words and $\boldsymbol{\pi}_\ell$ represents the POS tags of a target seen in an exemplar or training sentence as evoking f.

Some features involving WordNet relationships shown in Table 5.5 can be ablated to

check whether they hurt performance. If not and if this ablation study improves perforamce, it will indicate that word cluster information serve as a proxy for the lexical semantic relationships extracted from a limited lexical resource like WordNet, and it can be ignored as a mandatory lexical resource in our model for frame identification.

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 t_i in t exists, s.t. $t_i \in c_s$	● some word t_i in t exists, s.t. $t_i \in c_c$
● some word t_i in t exists, s.t. $t_i \in c_t$	
◐ some word t_i in t exists, s.t. $t_i \in c_s$, and the sentence uses PASSIVE voice	◐ some word t_i in t exists, s.t. $t_i \in c_s$, and the sentence uses ACTIVE voice
Span content features: apply to overt argument candidates.	
○ For some word w_{s_j} in s, $w_{s_j} \in c_c$	○ For some word w_{s_j} in s, $w_{s_j} \in c_t$
○ the head word of $s \in c_c$	○ the head word of $s \in c_t$
◐ For some word w_{s_j} in s, $w_{s_j} \in c_s$	◐ For some word w_{s_j} in s, $w_{s_j} \in c_s$, the voice is denoted in the span, s's position with respect to t (BEFORE, AFTER, or OVERLAPPING)
◐ For some word w_{s_j} in s, $w_{s_j} \in c_s$, the voice denoted in the span (ACTIVE or PASSIVE)	
Span context POS features: for overt candidates, up to 6 of these features will be active.	
○ a word belonging to c_c occurs up to 3 words before the first word of s	○ a word belonging to c_c occurs up to 3 words after the last word of s
○ a word belonging to c_t occurs up to 3 words before the first word of s	○ a word belonging to c_t occurs up to 3 words after the last word of s

Table 6.2: Proposed set of features to be used for argument identification. Here, t denotes the target that evokes a semantic frame, s is the candidate span being considered as an argument, c_c denotes a particular categorical word cluster, c_t denotes a particular topical word cluster, and c_s denotes a particular synonymous word cluster. Rest of the notation follows from §5.4.

Taking inspiration from the features shown in Table 6.1, we can incorporate cluster information in the feature set for argument identification, which is shown in Table 5.7. A possible set of cluster-based features is tabulated in Table 6.2. These features incorporate all three types of cluster features: categorical, topical and synonymous. Due to the nature of these clusters, they will generalize to new data and reduce sparsity, and provide valuable signals extracted from unlabeled data to the log-linear model devised for argument identification, and could improve the performance of overall frame-semantic parsing.

We hope that the use of word clusters extracted from very large volumes of data like the Gigaword will help the model to generalize better to new data unlike supervised models trained on a small set of domains, e.g. the SemEval'07 data.

6.2.2 Word Clusters in Paraphrase Identification

Word cluster information in the form of cluster IDs can be incorporated into the paraphrase identification model described in Chapter 4 to either improve the model's performance or reduce the dependence on preprocessing tools like a part-of-speech tagger or a named-entity recognizer. It may also obviate the use of a lexical resource like WordNet. We call the proposed task of incorporating word clusters into our paraphrase model WC2. The paremeterization of our quasi-sychronous grammar model for identifying paraphrases occurred at the p_{kid} level, which we defined in Equations 4.6-4.12. For the ease of the reader, we reproduce it below:

$$
\begin{aligned}
& p_{kid}(t_i, \tau_{\mathrm{l}}^{\mathbf{t}}(i), a(i) \mid t_j, l, \tau^{\mathbf{s}}) = \\
& p_{\mathit{config}}(\mathit{config}(t_i, t_j, s_{a(i)}, s_l) \mid t_j, l, \tau^{\mathbf{s}}) && (6.1)\\
& \quad \times p_{\mathit{unif}}(a(i) \mid \mathit{config}(t_i, t_j, s_{a(i)}, s_l)) && (6.2)\\
& \quad \times p_{\mathit{lab}}(\tau_{\mathrm{l}}^{\mathbf{t}}(i) \mid \mathit{config}(t_i, t_j, s_{a(i)}, s_l)) && (6.3)\\
& \quad \times p_{pos}(pos(t_i) \mid pos(s_{a(i)})) && (6.4)\\
& \quad \times p_{ne}(ne(t_i) \mid ne(s_{a(i)})) && (6.5)\\
& \quad \times p_{\mathit{lsrel}}(\mathit{lsrel}(t_i) \mid s_{a(i)}) && (6.6)\\
& \quad \times p_{word}(t_i \mid \mathit{lsrel}(t_i), s_{a(i)}) && (6.7)
\end{aligned}
$$

Word cluster information can be incorporated in the previous equation at several locations. Some of these possibilities are enumerated below.

P1. p_{pos} (expression 6.4) can be converted to a log-linear model that defines a conditional probability distribution over the POS tag $pos(t_i)$ as well as the categorical word cluster $\mathcal{C}_c(t_i)$ observed at t_i, given the POS tag $pos(s_{a(i)})$ as well as the categorical word cluster $\mathcal{C}_c(s_{a(i)})$ observed at $s_{a(i)}$.

P2. p_{pos} can be redefined to be a log-linear model that does not use POS tags at all, and just uses categorical cluster information. If this works, then we reduce our dependence on a part-of-speech tagger.

P3. p_{ne} (expression 6.5) can be converted to a log-linear model that defines a conditional probability distribution over the NE tag $ne(t_i)$ as well as the categorical word cluster $\mathcal{C}_c(t_i)$ observed at t_i, given the POS tag $ne(s_{a(i)})$ as well as the categorical word cluster $\mathcal{C}_c(s_{a(i)})$ observed at $s_{a(i)}$.

P4. p_{ne} can be ignored completely, if either P1, P2 works. This is a possibility because word clusters often cluster named entities together. This would reduce our dependence on a named-entity recognizer.

P5. p_{pos}, p_{ne} and categorical word cluster information can be combined to one single log-linear model that models $pos(t_i)$, $ne(t_i)$ and $\mathcal{C}_c(t_i)$ given $pos(s_{a(i)})$, $ne(s_{a(i)})$ and $\mathcal{C}_c(s_{a(i)})$.

P6. Finally, expressions 6.6 and 6.7, which are components that rely on WordNet lookups, can be replaced with a log-linear model that models the probability of observing t_i given the synonymous cluster information $\mathcal{C}_s(s_{a(i)})$ of $s_{a(i)}$. This model may completely ignore WordNet lookups because we noticed during feature ablation studies that only the IDENTICAL WORD feature from WordNet (see §3.4) played an important role. If proper synonyms are being gathered using syntactic distributional similarity based clustering, such a log-linear model may eliminate our need to use WordNet for finding lexical semantic similarity.

Using word cluster information in the paraphrase model will reduce dependence on preprocessors and lexical resources, help in the generalization of the model to newer domains and will reduce the number of parameters in the model with efficient feature engineering.

6.3 Argument Identification with Phrasal Paraphrases

The FramNet lexicon contains 7124 roles and the training data for SemEval'07 contains 22093 role instances. This statistic suggests that there are very few unique spans corresponding to each role type in the lexicon, making the features used for argument identification very sparse. Several problems that deal with text units at the level of phrases face this kind of sparsity problem. Recently, Bannard and Callison-Burch (2005) investigated techniques for automatically extracting phrasal paraphrases from bilingual corpora that are traditionally used for machine translation. They demonstrated that for low resource language pairs, with the use of these extracted paraphrases, data sparsity for statistical machine translation can be reduced and better quality translation can be produced (Callison-Burch et al., 2006). More recently, Callison-Burch (2008) added syntactic constraints to his paraphrase extraction method, and released publicly available software to extract paraphrases.[1]

Lin and Wu (2009) also extracted phrase clusters or groups of categorically or topically similar phrases from huge volumes of data using the K-Means algorithm, and used these clusters as features in an information extraction task. In our proposed work, which we label as WC3, we plan to improve argument identification using phrasal paraphrases and phrase clusters. To extract phrase clusters, we will apply the K-Means++ algorithm to phrases that appear in query logs[2] and use the Gigaword corpus in the same way as we used it to find the word clusters, resulting in topical and categorical clusters. For phrasal paraphrases, we will use the software from Callison-Burch (2008). For argument identification, we plan to use these phrasal paraphrases and phrase clusters in the form of features, which we enumerate in Table 6.3.

Span content features: apply to overt argument candidates.
○ Span s belongs to categorical phrase cluster c^p_c
○ Span s belongs to topical phrase cluster c^p_t
◐ A real valued feature denoting the highest paraphrase similarity score between span s and a exemplar phrase associated with role r
◐ Span s belongs to topical phrase cluster c^p_t, the voice is denoted in the span, s's position with respect to t (BEFORE, AFTER, or OVERLAPPING)
◐ Span s belongs to topical phrase cluster c^p_t, the voice denoted in the span (ACTIVE or PASSIVE)

Table 6.3: Proposed set of features leveraging phrasal paraphrases and phrase clusters to be used for argument identification. Notation follows from §5.4. The third feature is a real valued feature and is computed using a paraphrase score between a span and all the phrases filling a role r in the exemplar sentences of FrameNet or the training data.

Using the features tabulated in Table 6.3, it may be possible to reduce data sparsity because of the miniscule size of the training corpus, and reduce argument identification errors.

[1] See `http://www.cs.jhu.edu/~ccb/howto-extract-paraphrases.html`

[2] There are several publicly available query logs. One example is the AOL dataset available at `http://brie.di.unipi.it/smalltext/datasets.html`.

6.4 Semi-supervised Learning for Frame Semantics

In this section, we focus on future work that investigates semi-supervised modeling techniques to improve the frame-semantic structure prediction task. Unlike the proposed work discussed in §6.2 and §6.3 that focuses on designing features based on clusters realized from unlabeled data, in the following subsections we explore directions that focus on the modeling technique, incorporating labeled as well as unlabeled data. Specifically, we discuss two kinds of semi-supervised learning approaches, that have been moderately investigated by the NLP community, resulting in improvements for information extraction and syntactic parsing tasks. These two approaches are entropy minimization (§6.4.1) and techniques for combining discriminative and generative models (§6.4.2. These techniques are perfectly suitable to our problem because our models are manifestations of conditional log-linear models at which these semi-supervised techniques are targeted. Moreoever, the frame-semantic parsing task has very little labeled data, and large volumes of unlabeled data leveraged using these semi-supervised extensions could improve the quality of semantic analysis.

6.4.1 Entropy Minimization for Frame-Semantic Parsing

Entropy minimization techniques for semi-supervised learning has been proposed by Grandvalet and Bengio (2004). Later on, the same framework has been adapted in NLP for the case of log-linear sequence labeling models like CRFs (Jiao et al., 2006) and for dependency parsing (Smith and Eisner, 2007). In both cases, improvements were achieved over vanilla supervised methods with small amounts of labeled data. We hope to employ this method for frame-semantic parsing and label this proposed task as SS1. The entropy minimization technique is a straightforward extension to the conditional maximum-likelihood criterion for log-linear models. For a supervised log-linear model, we maximize the following regularized conditional likelihood of the data (see §3.2 for basic details):

$$SL(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(y^{(i)} \mid x^{(i)}) - C\|\boldsymbol{\theta}\|_2^2 \tag{6.8}$$

The entropy minimization framework presented by Grandvalet and Bengio (2004) modifies the above expression to account for unlabeled data in the following way:

$$\begin{aligned} SSL(\boldsymbol{\theta}) &= \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(y^{(i)} \mid x^{(i)}) - C\|\boldsymbol{\theta}\|_2^2 \\ &+ \gamma \sum_{i=N+1}^{M} \sum_{y} p_{\boldsymbol{\theta}}(y \mid x^{(i)}) \log p_{\boldsymbol{\theta}}(y \mid x^{(i)}) \end{aligned} \tag{6.9}$$

The third term on the right hand side of Equation 6.9 denotes the negative conditional entropy of the unlabeled data $\mathcal{D}^u$ under the conditional log-linear model. γ is a positive real valued number that controls how much to trust the unlabeled data. This approach is motivated by the fact that minimizing the conditional entropy of unlabeled data encourages the algorithm to find potential labelings for the unlabeled data that are mutually reinforcing with the supervised labels; greater certainty on these potential labelings coincides with greater conditional likelihood on the supervised labels, and vice versa. It has been shown

empirically that this method partitions the unlabeled data into useful clusters. For example, experiments on finding identifying gene and protein mentions in biomedical text (Jiao et al., 2006) have shown that considerable improvements over purely supervised methods are possible using this method.

Note that the objective function in Equation 6.9 is non-convex. Hence, numerical optimization methods will lead to a local minumum. Moreoever, the challenge in this method lies in the fact that numerical optimization methods to optimize this objective function requires us to take its derivative with respect to the parameter vector $\boldsymbol{\theta}$. The derivative of the first two terms on the right hand size of Equation 6.9 is the same as a typical regularized conditional log-linear model (see Equation 3.5). However, the derivative of the negative conditional entropy comes out to be:

$$\frac{\partial}{\partial \theta_m}\Bigg(\sum_{i=N+1}^{M}\sum_{y} p_{\boldsymbol{\theta}}(y \mid x^{(i)}) \log p_{\boldsymbol{\theta}}(y \mid x^{(i)})\Bigg) = \sum_{i=N+1}^{M}\Bigg(\sum_{n}\theta_n\Big[\sum_{y} p_{\boldsymbol{\theta}}(y \mid x^{(i)}) f_n(x^{(i)}, y) f_m(x^{(i)}, y) - \big[\sum_{y} p_{\boldsymbol{\theta}}(y \mid x^{(i)}) f_n(x^{(i)}, y)\big]\big[\sum_{y} p_{\boldsymbol{\theta}}(y \mid x^{(i)}) f_m(x^{(i)}, y)\big]\Big]\Bigg)$$

Computing this derivative can be very expensive for large models with a lot of classes. For structured models like CRFs (Lafferty et al., 2001), this can be prohibitively expensive. Our argument identification model, which is a simple log-linear model with polynomial number of labels, can be extended with an entropy regularizer as described above, and the expense is not as large as compared to a model with exponential number of labels. However, there are a certain number of challenges that need to be overcome before we can use this framework of semi-supervised learning. The argument identification model described in §5.4 assumes that the targets invoking semantic frames and the frames themselves are assumed to be given before the model is used to discover the semantic roles. For completely unlabeled data therefore, we need to first label potential targets and their semantic frames in some way, before we can find a data point's total entropy by summing up over all possible spans for a particular role.

To this end, first, we plan to use our target identification model described in §5.2 which has an F_1 score of 79% on the test data suggesting that it is moderately accurate. We assume that the noise in labeling wrong targets will be turned down by large amounts of unlabeled data. Next, we plan to run our frame identification model on these labeled targets, but only choose the frames on which our model has high confidence; here, confidence will be measured using the posterior probability assigned to a frame, given the model's observations. This measure of high confidence will be tuned on a development set, and then fixed for unlabeled data. We have the option of summing up over all possible frames for a target and treat the frame as unobserved, but estimating model parameters for that option will be very expensive. Finally, we would run the entropy minimization technique to hope that it would improve the argument identification model.

For a log-linear model that incorporates a hidden variable, the conditional log-likelihood of the data has the following form:

$$SHL(\boldsymbol{\theta}) = \sum_{i=1}^{N} \underbrace{\left[\sum_{z} \boldsymbol{\theta}^\top f(x^{(i)}, z, y^{(i)}) - \log \sum_{y} \sum_{z'} \exp \boldsymbol{\theta}^\top f(x^{(i)}, z', y) \right]}_{\log p'(y^{(i)} | x^{(i)})} - C\|\boldsymbol{\theta}\|_2^2 \tag{6.10}$$

This model is exactly similar to the model we have described for frame identification in §5.3. Note that because of the presence of the hidden variable, computing derivatives for this model is more expensive than the supervised model. The negative conditional entropy of unlabeled data $\mathcal{D}^u$ for this model will look like:

$$\begin{aligned} -Entropy(\mathcal{D}^u) = & \sum_{i=1}^{N} \sum_{y'} p'(y' \mid x^{(i)}) \times \\ & \left[\sum_{z} \boldsymbol{\theta}^\top f(x^{(i)}, z, y') - \log \sum_{y} \sum_{z'} \exp \boldsymbol{\theta}^\top f(x^{(i)}, z', y) \right] \end{aligned} \tag{6.11}$$

Computing the derivative of this term will involve a specific set of latent variables for each semantic frame (the variables y and y'), and thus will take up a complex form.

For training a frame identification model using entropy minimization, we first intend to label the unlabeled data with targets, by labeling *all* content words. Ideally in frame semantics, every content word can evoke a frame, although this was not followed for the data released in the SemEval'07 task, where targets containing the maximum semantic information in a sentence were selectively labeled. Labeling every content word as a candidate target in the unlabeled data would enable the model to find frame identification features for as many targets as possible; when presented with gold targets on real test data, we hope that the model will be able to find better frames for targets unseen in the FrameNet lexicon and supervised training data.

As an alternate strategy, we will also attempt to use our target identification heuristics (§5.2) instead of labeling all content words, and follow the optimization procedure for the entropy minimization framework on the combined set of labeled and unlabeled data.

6.4.2 Mixture of Discriminative and Generative Models

In this section, we consider the use of models that efficiently include parameters estimated using generative models into conditional log-linear models, and plan to use these for frame-semantic parsing. We call this proposed direction of work SS2. Suzuki and Isozaki (2008) proposed a model that blended several weighted generative models into the feature function of a conditional log-linear model. The parameters of the generative models were estimated on unlabeled data. They used their models on a sequence labeling task and used very large-scale unlabeled data for their experiments, and achieved improvements over supervised methods. Recently, Suzuki et al. (2009) used the same technique for dependency parsing, to get improvements over the corresponding supervised setting.

The extension of a conditional log-linear model to make room for generative models is straightforward. The conditional probability under such a model looks like the following:

$$p_{\boldsymbol{\theta}}(y \mid x) = \frac{\exp g(x, y)}{\sum_{y'} \exp g(x, y')} \tag{6.12}$$

where, g is a function that maps the observation/label pair (x, y) to a non-negative real number. For a traditional log-linear model,

$$g(x, y) = \boldsymbol{\theta}^\top \mathbf{f}(x, y) \tag{6.13}$$

Here, like before, $\boldsymbol{\theta}$ are the model parameters and $\mathbf{f}$ is a vector of features. To integrate a generative component into such a model, we modify the function g as follows:

$$g(x, y) = \boldsymbol{\theta}^\top \mathbf{f}(x, y) + \boldsymbol{\varsigma}^\top \mathbf{q}(x, y) \tag{6.14}$$

Here, $\mathbf{q}$ is a k-dimensional vector, whose components q_j are functions trained on unlabeled data, and $\boldsymbol{\varsigma}$ are relative strengths of each function q_j and are trained on labeled data. Also let $\boldsymbol{\Theta}$ denote the concatenation of $\boldsymbol{\theta}$ and $\boldsymbol{\varsigma}$, thus defining a conditional distribution $p_{\boldsymbol{\Theta}}(y \mid x)$. To derive $\mathbf{q}$, the following is done. The actual feature vector is partitioned into k disjoint parts, resulting in k functions $\mathbf{r}_1(x, y) \ldots \mathbf{r}_k(x, y)$. Next the following generative probability is defined:

$$q'_j(x, y) = \prod_{d=1}^{D_j} \beta_{j,d}^{r_{j,d}(x,y)} \tag{6.15}$$

Here, D_j is the number of dimensions of $\mathbf{r}_j(x, y)$ and $\beta_{j,1} \ldots \beta_{j,D_j}$, the parameters of this model form a multinomial distribution such that $\beta_{j,d} \geq 0$ and $\sum_{d=1}^{D_j} \beta_{j,d} = 1$. Next, q_j are just defined as logarithms of q'_j:

$$q_j(x, y) = \sum_{d=1}^{D_j} r_{j,d}(x, y) \log \beta_{j,d} \tag{6.16}$$

The parameters $\beta_{j,d}$ of the smaller generative models are estimated using a lot of unlabeled data. For unlabeled data, the log-likelihood of the data under the j^{th} generative model is defined to be:

$$\sum_{i=1}^{M} \sum_{y} p_{\boldsymbol{\Theta}}(y \mid x) \log q'_j(x, y) \tag{6.17}$$

The above expression resembles the likelihood expression for the M-step of the EM algorithm, with the only difference that the posterior probability of the data $p_{\boldsymbol{\Theta}}(y \mid x)$ (a log-linear model with the exponent taking the form shown in Equation 6.14) is used instead of the standard Q function.[3] To estimate the parameters $\beta_{j,d}$ appearing in expression 6.17, a renormalization procedure like the M-step is followed.

For complete parameter estimation, an iterative procedure is followed. At first, $\forall j$, the parameters $\beta_{j,d}$ in the j^{th} generative model are initialized such that it follows a uniform distribution. With $\beta_{j,d}$ fixed, the parameters $\boldsymbol{\Theta}$ of the log-linear model, that contain the standard feature weights $\boldsymbol{\theta}$ as well as the weights $\boldsymbol{\varsigma}$ of the q_j functions, are trained on the labeled data using a numerical optimization method like L-BFGS. Next, at the second step, by fixing $\boldsymbol{\Theta}$, the parameters in the generative models are trained by maximizing the expression 6.17 on unlabeled data. Finally, with the trained $\beta_{j,d}$ fixed, another step of parameter training

[3]The standard form of likelihood for the M-step of the EM algorithm is as follows: $\sum_{x,y} \tilde{p}(x) Q(y \mid x) \log p(x, y)$. Here, $Q(y \mid x)$ is a distribution over labels given observations that is estimated using the E-step.

for the log-linear model is done on supervised data to estimate $\boldsymbol{\Theta}$. This iterative training procedure can be optionally done several times alternating between training the log-linear model and training the generative models.

Both probabilistic models for the frame-semantic parsing task are amenable to this kind of a training procedure. The challenging steps required to adapt our models requires us to partition the feature set into several smaller generative models meaningfully. For the dependency parsing task (Suzuki et al., 2009), 140 such partitions were found using careful partitioning. Another important modification to this algorithm for the frame identification task will require us to adapt the framework to a log-linear model containing hidden variables. Moreover, the challenges mentioned in §6.4.1 about not having gold targets for the frame identifiation task and not having gold targets and frames for the argument identification task prevails for this kind of semi-supervised training too. We plan to follow similar approaches as mentioned in §6.4.1 to handle the absence of gold labels for the unlabeled data.

We hope to gain benefits by using the semi-supervised techniques considered in this section because our problem of frame-semantic parsing boasts very little training data, and our existing feature rich probabilistic frameworks should benefit by leveraging unlabeled data by either minimizing its entropy or finding generative model parameters describing it.

Chapter 7

Conclusion

We have provided detailed descriptions of probabilistic methods used to predict structures useful to analyze natural language semantics. We have focused on two different aspects of semantic processing – paraphrase modeling and prediction of frame-semantic structures. For both tasks, our models have been able to achieve state-of-the-art results, and for frame-semantic analysis, we have made available a tool for the natural language processing community to use.

A major focus in our models has been in the use of modeling latent structures useful to describe some semantic phenomenon, but unavailable as annotations in labeled corpora. We have used latent variables to model word alignments in monolingual sentence pairs and to model latent lexical units for semantic frames.

Whenever required, we have made use of distributed computing to expedite the process of parameter learning for large scale models, and for deriving useful feature information from large corpora.

As future work, we have proposed the use of very large corpora along with available annotated data. Our proposed work includes the derivation of features from unlabeled data, as well as the use of unlabeled data for parameter learning in a semi-supervised framework. The goal of this thesis is to build robust computational models for natural language semantics, for which we have developed initial probabilistic frameworks. Our hope is to demonstrate that with the use of unannotated data in these frameworks, we will be able to harvest useful semantic information and significantly improve the quality of semantic analysis of text.

The following section looks as a tentative schedule for the proposed work.

7.1 Timeline of Proposed Work

Table 7.1 shows an estimated schedule of the proposed work. The table also shows some required tasks to be completed as a part of a doctoral degree.

Semester	Task	Description of Task
Spring 2010	WC1 §6.2.1	Incorporating word clusters in frame-semantic parsing
	Writing	Writing paper for EMNLP 2010
Summer 2010	Other	Internship at Google Research, New York on semi-supervised learning for NLP.
Fall 2010	WC2 §6.2.2	Incorporating word clusters in paraphrase identification
	WC3 §6.3	Incorporating phrase clusters and paraphrases in argument identification for frame-semantic parsing
	SS1 §6.4.1	Initial work on entropy minimization techniques for frame-semantic parsing.
	Other	Side project on Arabic syntax processing
	Other	Mandatory teaching assistantship
Spring 2011	SS1 §6.4.1	Entropy minimization techniques for frame-semantic parsing.
	SS2 §6.4.2	Initial work on semi-supervised mixture of discriminative and generative models for frame-semantic parsing.
	Writing	Writing paper for ACL 2011
Summer 2011	SS2 §6.4.2	Semi-supervised mixture of discriminative and generative models for frame-semantic parsing.
	Writing	Writing paper for EMNLP 2011
Fall 2011	Writing	Writing Dissertation
	Other	Job search
	Defense	November 2011

Table 7.1: Timeline for proposed work

Bibliography

Al-Onaizan, Y., Curin, J., Jahr, M., Knight, K., Lafferty, J., Melamed, I. D., Och, F., Purdy, D., Smith, N., and Yarowsky, D. (1999). Statistical machine translation. In *Final Report, JHU Workshop*. [2]

Arthur, D. and Vassilvitskii, S. (2007). k-means++: the advantages of careful seeding. In *Proc. of SODA*. [22]

Baker, C., Ellsworth, M., and Erk, K. (2007). SemEval-2007 Task 19: Frame semantic structure extraction. In *Proc. of SemEval*. [12, 38, 41]

Bannard, C. and Callison-Burch, C. (2005). Paraphrasing with bilingual parallel corpora. In *Proc. of ACL*. [9, 59]

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 *Proc. of the Second PASCAL Recognising Textual Entailment Challenge*. [1, 8]

Barzilay, R. and Lee, L. (2003). Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. In *Proc. of NAACL*. [9, 25]

Barzilay, R. and McKeown, K. R. (2001). Extracting paraphrases from a parallel corpus. In *Proc. of ACL*. [9]

Berger, A. (1996). A brief maxent tutorial. [17]

Bikel, D. M., Schwartz, R., and Weischedel, R. M. (1999). An algorithm that learns what's in a name. *Machine Learning*, 34(1). [28]

Bilotti, M. W., Ogilvie, P., Callan, J., and Nyberg, E. (2007). Structured retrieval for question answering. In *Proc. of SIGIR*. [5]

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proc. of COLT*. [54]

Boas, H. C. (2002). Bilingual framenet dictionaries for machine translation. In *Proc. of LREC*. [13]

Bos, J. and Markert, K. (2005). Recognising textual entailment with logical inference. In *Proc. of HLT*. [1, 5, 9]

Bottou, L. (2003). Stochastic learning. In *Advanced Lectures on Machine Learning*. [18, 48]

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). [22]

Brown, P. F., Pietra, V. J. D., Pietra, V. J. D., and Mercer, R. L. (1991). Word-sense disambiguation using statistical methods. In *Proc. of ACL*. [1, 2]

Bruce, R. F. and Wiebe, J. (1994). Word-sense disambiguation using decomposable models. In *Proc. of ACL*. [2]

Burchardt, A. (2006). Approaching textual entailment with LFG and FrameNet frames. In *Proc. of the Second PASCAL RTE Challenge Workshop*. [13]

Burchardt, A., Erk, K., and Frank, A. (2005). A WordNet detour to FrameNet. In *Sprachtechnologie, mobile Kommunikation und linguistische Resourcen*. [44]

Burchardt, A., Pennacchiotti, M., Thater, S., and Pinkal, M. (2009). Assessing the impact of frame semantics on textual entailment. *Natural Language Engineering*, 15. [13]

Callison-Burch, C. (2008). Syntactic constraints on paraphrases extracted from parallel corpora. In *Proc. of EMNLP*. [ii, 10, 59]

Callison-Burch, C., Koehn, P., and Osborne, M. (2006). Improved statistical machine translation using paraphrases. In *Proc. HLT-NAACL*. [6, 25, 59]

Carreras, X. and Màrquez, L. (2004). Introduction to the conll-2004 shared task: Semantic role labeling. In *Proc. of CoNLL*. [11]

Carreras, X. and Màrquez, L. (2005). Introduction to the conll-2005 shared task: semantic role labeling. In *Proc. of CoNLL*. [11]

Chang, M.-W., Ratinov, L., and Roth, D. (2007). Guiding semi-supervision with constraint-driven learning. In *Proc. of ACL*. [54]

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proc. of NAACL*. [2, 3]

Cohn, T. and Blunsom, P. (2005). Semantic role labelling with tree conditional random fields. In *Proc. of CoNLL*. [2]

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

Corley, C. and Mihalcea, R. (2005). Measuring the semantic similarity of texts. In *Proc. of ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*. [1, 10, 25]

Culotta, A. and Sorensen, J. (2004). Dependency tree kernels for relation extraction. In *Proc. of ACL*. [3]

Dagan, I., Glickman, O., and Magnini, B. (2005). The PASCAL recognising textual entailment challenge. In *Proc. of MLCW*. [1, 8]

Das, D., Kumar, M., and Rudnicky, A. I. (2008). Automatic extraction of briefing templates. In *Proc. of IJCNLP*. [5]

Das, D., Schneider, N., Chen, D., and Smith, N. A. (2010a). Probabilistic frame-semantic parsing. In *Proc. of NAACL-HLT*. [i, 38]

Das, D., Schneider, N., Chen, D., and Smith, N. A. (2010b). SEMAFOR 1.0: a probabilistic frame-semantic parser. Technical report, CMU-LTI-10-001. [38]

Das, D. and Smith, N. A. (2009). Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proc. of ACL-IJCNLP*. [i, 25]

Dean, J. and Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1). [19]

Deschacht, K. and Moens, M.-F. (2009). Semi-supervised semantic role labeling using the latent words language model. In *Proc. of EMNLP*. [53]

Dolan, B., Quirk, C., and Brockett, C. (2004). Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources. In *Proc. of COLING*. [10, 32]

Dolan, W. B. and Brockett, C. (2005). Automatically constructing a corpus of sentential paraphrases. In *Proc. of IWP*. [9, 32, 34]

Eisner, J. (1996). Three new probabilistic models for dependency parsing: An exploration. In *Proc. of COLING*. [16]

Erk, K. and Padó, S. (2006). Shalmaneser - a toolchain for shallow semantic parsing. In *Proc. of LREC*. [12]

Fellbaum, C., editor (1998). *WordNet: an electronic lexical database*. [i, 5, 21, 34, 41]

Fillmore, C. J. (1982). Frame Semantics. In *Linguistics in the Morning Calm*. [4, 8, 12, 37]

Fillmore, C. J., Johnson, C. R., and Petruck, M. R. (2003). Background to FrameNet. *International Journal of Lexicography*, 16(3). [i, 4, 12, 37]

Finch, A., Hwang, Y. S., and Sumita, E. (2005). Using machine translation evaluation techniques to determine sentence-level semantic equivalence. In *Proc. of IWP*. [10, 25]

Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. In *Proc. of ACL*. [2]

Fleischman, M., Kwon, N., and Hovy, E. (2003). Maximum entropy models for FrameNet classification. In *Proc. of EMNLP*. [12]

Fung, P. and Chen, B. (2004). BiFrameNet: bilingual frame semantics resource construction by cross-lingual induction. In *Proc. of COLING*. [13]

Fürstenau, H. and Lapata, M. (2009). Semi-supervised semantic role labeling. In *Proc. of EACL*. [13]

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

Giampiccolo, D., Magnini, B., Dagan, I., and Dolan, B. (2007). The third PASCAL recognizing textual entailment challenge. In *Proc. of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*. [1, 8]

Gildea, D. and Jurafsky, D. (2002). Automatic labeling of semantic roles. *Computational Linguistics*, 28(3). [2, 11, 12]

Gimpel, K. and Smith, N. A. (2009). Feature-rich translation by quasi-synchronous lattice parsing. In *Proc. of EMNLP*. [27]

Giuglea, A. and Moschitti, A. (2006). Shallow semantic sarsing based on FrameNet, VerbNet and PropBank. In *Proc. of ECAI 2006*. [12, 13]

Glickman, O. and Dagan, I. (2005). Web based probabilistic textual entailment. In *Proc. of PASCAL Challenge Workshop for Recognizing Textual Entailmen*. [2, 9]

Glickman, O., Dagan, I., and Koppel, M. (2005). A probabilistic classification approach for lexical textual entailment. In *Proc. of AAAI*. [1, 2]

Gorman, J. and Curran, J. R. (2006). Scaling distributional similarity to large corpora. In *Proc. of COLING-ACL*. [56]

Graff, D. (2003). English Gigaword. Linguistic Data Consortium. [22, 30, 54]

Grandvalet, Y. and Bengio, Y. (2004). Semi-supervised learning by entropy minimization. In *Proc. of NIPS*. [54, 60]

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

Haghighi, A., Liang, P., Berg-Kirkpatrick, T., and Klein, D. (2008). Learning bilingual lexicons from monolingual corpora. In *Proc. of ACL-HLT*. [2]

Harabagiu, S. and Hickl, A. (2006). Methods for using textual entailment in open-domain question answering. In *Proc. of COLING-ACL*. [6]

He, S. and Gildea, D. (2006). Integrating cluster information for cross-frame semantic role labeling integrating cluster information for cross-frame semantic role labeling integrating cluster information for cross-frame semantic role labeling. Technical Report 892, The University of Rochester. [53]

Heilman, M. and Smith, N. A. (2010). Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proc. of NAACL-HLT*. [11]

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800. [25]

Jebara, T. and Pentland, A. (1999). Maximum conditional likelihood via bound maximization and the cem algorithm. In *Proc. of NIPS*. [19]

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 *Proc. of ACL*. [ii, 54, 60]

Jijkoun, V. and de Rijke, M. (2005). Recognizing textual entailment using lexical similarity. In *Proc. of the PASCAL Challenges Workshop on Recognizing Textual Entailment*. [9]

Joachims, T. (1999). Making large-scale SVM learning practical. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press. [34]

Johansson, R. and Nugues, P. (2007). LTH: semantic structure extraction using nonprojective dependency trees. In *Proc. of SemEval*. [12, 13, 41, 42, 44]

Johansson, R. and Nugues, P. (2008). Dependency-based semantic role labeling of PropBank. In *Proc. of EMNLP*. [41]

Joshi, A. K. and Schabes, Y. (1997). Tree adjoining grammars. In Rozenberg, G. and Salomaa, K., editors, *Handbook of Formal Languages*. Springer. [54]

Kingsbury, P. and Palmer, M. (2002). From TreeBank to PropBank. In *Proc. of LREC*. [5, 11, 14]

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 *Proc. of ACL*. [2]

Kok, S. and Brockett, C. (2010). Hitting the right paraphrases in good time. In *Proc. of NAACL-HLT*. [10]

Koo, T., Carreras, X., and Collins, M. (2008). Simple semi-supervised dependency parsing. In *Proc. of ACL-HLT*. [i, 22, 53, 55]

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*. [61]

Lin, D. and Pantel, P. (2001). DIRT - discovery of inference rules from text. In *Proc. of KDD*. [9, 56]

Lin, D. and Wu, X. (2009). Phrase clustering for discriminative learning. In *Proc. of ACL-IJCNLP*. [i, 9, 22, 23, 53, 54, 55, 59]

Liu, D. C. and Nocedal, J. (1989). On the limited memory bfgs method for large scale optimization. *Math. Programming*, 45(3). [18]

MacCartney, B., Grenager, T., de Marneffe, M.-C., Cer, D., and Manning, C. D. (2006). Learning to recognize features of valid textual entailments. In *Proc. of HLT-NAACL*. [9]

MacCartney, B. and Manning, C. D. (2007). Natural logic for textual inference. In *Proc. of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*. [1, 9, 35]

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In *Proc. of the Berkeley Symposium on Mathematical Statistics and Probability*, volume 1. [22]

Malakasiotis, P. (2009). Paraphrase recognition using machine learning to combine similarity measures. In *Proc. of ACL Student Research Workshop*. [10]

Mann, G. S. and McCallum, A. (2007). Simple, robust, scalable semi-supervised learning via expectation regularization. In *Proc. of ICML*, pages 593–600, New York, NY, USA. ACM. [54]

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, 11, 14]

Màrquez, L., Carreras, X., Litkowski, K. C., and Stevenson, S. (2008). Semantic role labeling: an introduction to the special issue. *Computational Linguistics*, 34(2). [11]

Marsi, E. and Krahmer, E. (2005). Explorations in sentence fusion. In *Proc. of EWNLG*. [25]

Martins, A. F. T., Das, D., Smith, N. A., and Xing, E. P. (2008). Stacking dependency parsers. In *Proc. of EMNLP*. [3, 16]

Marton, Y., Callison-Burch, C., and Resnik, P. (2009). Improved statistical machine translation using monolingually-derived paraphrases. In *Proc. of EMNLP*. [6]

Matsubayashi, Y., Okazaki, N., and Tsujii, J. (2009). A comparative study on generalization of semantic roles in FrameNet. In *Proc. of ACL-IJCNLP*. [13, 48]

McClosky, D., Charniak, E., and Johnson, M. (2006). Effective self-training for parsing. In *Proc. of HLT-NAACL*. [54]

McDonald, R., Crammer, K., and Pereira, F. (2005). Online large-margin training of dependency parsers. In *Proc. of ACL*. [3, 16, 34]

McKeown, K. R. (1979). Paraphrasing using given and new information in a question-answer system. In *Proc. of ACL*. [9]

Melamed, I. D. (2004). Statistical machine translation by parsing. In *Proc. of ACL*. [27]

Meyers, A., Reeves, R., Macleod, C., Szekely, R., Zielinska, V., Young, B., and Grishman, R. (2004). The nombank project: An interim report. In *Proc. of NAACL/HLT Workshop on Frontiers in Corpus Annotation*. [12]

Miller, S., Guinness, J., and Zamanian, A. (2004). Name tagging with word clusters and discriminative training. In *Proc. of HLT-NAACL*. [i, 53]

Moschitti, A., Morărescu, P., and Harabagiu, S. M. (2003). Open-domain information extraction via automatic semantic labeling. In *Proc. of FLAIRS*. [13]

Narayanan, S. and Harabagiu, S. (2004). Question answering based on semantic structures. In *Proceedings of COLING*. [13]

Nivre, J., Hall, J., and Nilsson, J. (2004). Memory-based dependency parsing. In *Proceedings of CoNLL*. [3]

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*. [13]

Padó, S., Galley, M., Jurafsky, D., and Manning, C. D. (2009). Robust machine translation evaluation with entailment features. In *Proc. of ACL-IJCNLP*. [6]

Padó, S. and Lapata, M. (2005). Cross-linguistic projection of role-semantic information. In *Proc. of HLT-EMNLP*. [13]

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 *Proc. of ACL*. [5, 6, 34]

Pennacchiotti, M., Cao, D. D., Basili, R., Croce, D., and Roth, M. (2008). Automatic induction of FrameNet lexical units. In *Proc. of EMNLP*. [13]

Pereira, F., Tishby, N., and Lee, L. (1993). Distributional clustering of english words. In *Proc. of ACL*. [9, 53]

Petrov, S. and Klein, D. (2008). Sparse multi-scale grammars for discriminative latent variable parsing. In *Proc. of EMNLP*. [2, 19]

Pradhan, S. S., Ward, W. H., Hacioglu, K., Martin, J. H., and Jurafsky, D. (2004). Shallow semantic parsing using support vector machines. In *Proc. of HLT-NAACL*. [11]

Punyakanok, V., Roth, D., W.-T. Yih, and Zimak, D. (2004). Semantic role labeling via integer linear programming inference. In *Proc. of COLING*. [11]

Qiu, L., Kan, M.-Y., and Chua, T.-S. (2006). Paraphrase recognition via dissimilarity significance classification. In *Proc. of EMNLP*. [5, 10]

Quirk, C., Brockett, C., and Dolan, W. B. (2004). Monolingual machine translation for paraphrase generation. In *Proc. of EMNLP*. [10, 32]

Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging. In *Proc. of EMNLP*. [28, 34, 41]

Ravichandran, D. and Hovy, E. (2002). Learning surface text patterns for a question answering system. In *Proc. of ACL*. [9]

Ruppenhofer, J., Ellsworth, M., Petruck, M. R. L., Johnson, C. R., and Scheffczyk, J. (2006). FrameNet II: extended theory and practice. [48]

Schuler, K. K. (2005). *Verbnet: a broad-coverage, comprehensive verb lexicon*. PhD thesis, University of Pennsylvania. [12]

Shen, D. and Lapata, M. (2007). Using semantic roles to improve question answering. In *Proc. of EMNLP-CoNLL*. [5, 13]

Shi, L. and Mihalcea, R. (2004). An algorithm for open text semantic parsing. In *Proc. of Workshop on Robust Methods in Analysis of Natural Language Data*. [12]

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*. [13]

Smith, D. A. and Eisner, J. (2006). Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proc. of the HLT-NAACL Workshop on Statistical Machine Translation*. [10, 25, 27, 29]

Smith, D. A. and Eisner, J. (2007). Bootstrapping feature-rich dependency parsers with entropic priors. In *Proc. of EMNLP*. [ii, 60]

Smith, D. A. and Eisner, J. (2009). Parser adaptation and projection with quasi-synchronous grammar features. In *Proc. of EMNLP*. [27]

Smith, D. A. and Smith, N. A. (2007). Probabilistic models of nonprojective dependency

trees. In *Proc. of EMNLP-CoNLL*. [2]

Smith, N. A. (2006). *Novel estimation methods for unsupervised discovery of latent structure in natural language text*. PhD thesis, Johns Hopkins University. [2]

Snow, R., Jurafsky, D., and Ng, A. Y. (2006). Semantic taxonomy induction from heterogenous evidence. In *Proc. of COLING-ACL*. [2]

Steedman, M., Osborne, M., Sarkar, A., Clark, S., Hwa, R., Hockenmaier, J., Ruhlen, P., Baker, S., and Crim, J. (2003). Bootstrapping statistical parsers from small datasets. In *Proc. of EACL*. [54]

Surdeanu, M., Harabagiu, S., Williams, J., and Aarseth, P. (2003). Using predicate-argument structures for information extraction. In *Proc. of ACL*. [13]

Suzuki, J. and Isozaki, H. (2008). Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In *Proc. of ACL-HLT*. [ii, 55, 62]

Suzuki, J., Isozaki, H., Carreras, X., and Collins, M. (2009). An empirical study of semi-supervised structured conditional models for dependency parsing. In *Proc. of EMNLP*. [ii, 62, 63]

Thompson, C. A., Levy, R., and Manning, C. D. (2003). A generative model for semantic role labeling. In *Proc. of ECML*. [12]

Tjong Kim Sang, E. F. (2002). Introduction to the conll-2002 shared task: language-independent named entity recognition. In *Proc. of CoNLL*. [2]

Tjong Kim Sang, E. F. and Buchholz, S. (2000). Introduction to the conll-2000 shared task: chunking. In *Proc. of CoNLL*. [2]

Toutanova, K., Haghighi, A., and Manning, C. (2005). Joint learning improves semantic role labeling. In *Proc. of ACL*. [11, 48]

Turian, J. P., Shen, L., and Melamed, I. D. (2003). Evaluation of machine translation and its evaluation. In *Proc. of Machine Translation Summit IX*. [34]

Vapnik, V. N. (1995). *The nature of statistical learning theory*. Springer. [10]

Wan, S., Dras, M., Dale, R., and Paris, C. (2006). Using dependency-based features to take the "para-farce" out of paraphrase. In *Proc. of ALTW*. [10, 25, 34]

Wang, M., Smith, N. A., and Mitamura, T. (2007). What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proc. of EMNLP-CoNLL*. [3, 5, 25, 27, 29]

Wu, D. (1997). Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Comput. Linguist.*, 23(3). [11]

Wu, D. (2005). Recognizing paraphrases and textual entailment using inversion transduction grammars. In *Proc. of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*. [11]

Wu, D. and Fung, P. (2009). Semantic roles for smt: a hybrid two-pass model. In *Proc. of NAACL*. [5]

Yamada, H. and Matsumoto, Y. (2003). Statistical dependency analysis with support vector machines. In *Proc. of IWPT*. [16, 28]

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. of ACL*. [1, 2]

Yi, S.-T., Loper, E., and Palmer, M. (2007). Can semantic roles generalize across genres? In *Proc. of HLT-NAACL*. [1, 12]

Zettlemoyer, L. S. and Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proc. of UAI*. [1]

Zhang, Y. and Patrick, J. (2005). Paraphrase identification by text canonicalization. In *Proc. of ALTW*. [10, 25]

Zollmann, A. and Venugopal, A. (2006). Syntax augmented machine translation via chart parsing. In *Proc. of WMT*. [3]