SEMANTIC ROLE LABELING USING LEXICALIZED TREE ADJOINING GRAMMARS

by

Yudong Liu

BSc, Jilin University, China, 1999

MSc, Jilin University, China, 2002

a Thesis submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
Fall 2009

APPROVAL

Name: Yudong Liu

Degree: Doctor of Philosophy

Title of Thesis: Semantic Role Labeling using Lexicalized Tree Adjoining Grammars

Examining Committee: Dr. James P. Delgrande
Chair

Dr. Anoop Sarkar, Senior Supervisor

Dr. Evgenia (Eugenia) Ternovska Supervisor

Dr. Fred Popowich, SFU Examiner

Dr. Kristina N. Toutanova, External Examiner,
Researcher, Natural Language Processing Group,
Microsoft Research

Date Approved: _____

Abstract

The predicate-argument structure (PAS) of a natural language sentence is a useful representation that can be used for a deeper analysis of the underlying meaning of the sentence or directly used in various natural language processing (NLP) applications. The task of semantic role labeling (SRL) is to identify the predicate-argument structures and label the relations between the predicate and each of its arguments. Researchers have been studying SRL as a machine learning problem in the past six years, after large-scale semantically annotated corpora such as FrameNet and PropBank were released to the research community. Lexicalized Tree Adjoining Grammars (LTAGs), a tree rewriting formalism, are often a convenient representation for capturing locality of predicate-argument relations.

Our work in this thesis is focused on the development and learning of the state of the art discriminative SRL systems with LTAGs. Our contributions to this field include:

We apply to the SRL task a variant of the LTAG formalism called LTAG-spinal and the associated LTAG-spinal Treebank (the formalism and the Treebank were created by Libin Shen). Predicate-argument relations that are either implicit or absent from the original Penn Treebank are made explicit and accessible in the LTAG-spinal Treebank, which we show to be a useful resource for SRL.

We propose the use of the LTAGs as an important additional source of features for the SRL task. Our experiments show that, compared with the best-known set of features that are used in state of the art SRL systems, LTAG-based features can improve SRL performance significantly.

We treat multiple LTAG derivation trees as latent features for SRL and introduce a novel learning framework – Latent Support Vector Machines (LSVMs) to the SRL task using these latent features. This method significantly outperforms state of the art SRL

systems.

In addition, we adapt an SRL framework to a real-world ternary relation extraction task in the biomedical domain. Our experiments show that the use of SRL related features significantly improves performance over the system using only shallow word-based features.

Keywords: Semantic role labeling, lexicalized tree adjoining grammars, LTAG features, latent support vector machines

Acknowledgments

First I would like to thank my senior supervisor Dr. Anoop Sarkar for his continuous support and guidance in my study at Natural Language Lab at SFU.

Many thanks to my dissertation committee members, Dr. Eugenia Ternovska and Dr. Fred Popowich for their valuable advice and suggestions in different phases of my PhD work and helping me to technically accomplish this study.

I am grateful to Gholamreza Haffari for his insightful discussion in the past year. His pursuit of research always encourages me. I appreciate the time I spent with other people in the NLP group at SFU. They are Zhongmin Shi, Yang Wendy Wang, Chris Demwell, Dong Song, Julia Birke, Mehdi M. Kashani, Maxim Roy, Baskaran Sankaran, Winona Wu, Milan Tofiloski and many other people not named here. I would say that I learnt a lot from them.

Thanks also to my dear friends who are always my good companions and being there for me. They are George Lee, Xin Carol Wu and Cherry Bai. And I cannot forget my other friends who shine through my years in Vancouver by their encouragement and care. They are James Yu, Jenny Pan, Yinan Zhang, Lei Duan, Dana Zhu, Jinyun Ren, Xiaohong Zhao and many other people.

Last but not the least, I would like to give the biggest thanks to my family – my parents Daozhi Liu and Xiuqin Cui and my two sisters Yuzhe Liu and Yuhui Liu. It is their unconditional love, endless support and patience that walks me to this point of the long journey.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter, we will give a background on semantic role labeling (SRL) and Lexicalized Tree Adjoining Grammars (LTAGs).

1.1 Semantic Role Labeling (SRL) and its Applications

One of the ultimate goals of Natural Language Processing (NLP) is natural language understanding. As an important intermediate step towards natural language understanding, semantic role labeling (SRL) plays a key role in many NLP applications. In this section, we will give a description of the SRL task and how it is related to a variety of NLP applications.

1.1.1 What is Semantic Role Labeling

Informally, semantic role labeling is the process of assigning a simple WHO did WHAT to WHOM, WHEN, WHERE, WHY, HOW, etc. structure to a sentence in text. For example,

(1) *Mary hit Jack with a ball yesterday.*
(2) *Jack was hit by Mary yesterday with a ball.*

We can see that *Mary, Jack, a ball* and *yesterday* play semantic roles, such as "*Hitter, Thing hit, Instrument, Temporal adjunct*", in both sentences. The task of SRL is to assign pre-defined semantic roles to these phrases regardless of their presence in different syntactic realizations. This layer of information is vital for us to fully understand the meaning of

the two sentences.

More generally, the example actually illustrates the phenomenon of *alternations of a verb class* (Levin, 1993). This alternation in syntactic realization of semantic argument is widespread, affecting most English verbs in some way, and the patterns exhibited by specific verbs vary widely. In addition, while the rapid advances of syntactic parsing techniques in recent years have had an enormous impact on A breadth of NLP applications, we still have a long way to go from syntactic analysis to full understanding of the meaning of a sentence. All this motivates researchers to develop an automatic and accurate technique for layering semantics on top of syntactic analysis, and to take important steps towards the ultimate goal of language understanding.

Semantic role labeling is a well-defined task in different annotation frameworks and it is attracting much research attention. SRL aims to identify and label all the arguments (or semantic roles) for each predicate occurring in a sentence. Specifically, it involves identifying constituents in the sentence that represent the predicate's arguments and assigning pre-specified semantic roles to them. Here are some examples with semantic role labels.

(3) *[$_{Agent}$Mary]* ***hit*** *[$_{Theme}$ Jack] [$_{Instrument}$ with a ball] [$_{Temporal}$yesterday].*

(4) *[$_{Theme}$Jack] was* ***hit*** *by [$_{Agent}$ Mary] [$_{Temporal}$yesterday] [$_{Instrument}$ with a ball].*

(5) *[$_{A0}$John]* ***broke*** *[$_{A1}$ the window].*

(6) *[$_{A1}$The window] was* ***broken*** *by [$_{A0}$ John].*

In these examples, the subscripted information represents the semantic role labels that are assigned to the arguments of predicate (in bold face). (3) and (4) are examples of SRL annotation defined in FrameNet Scheme (Fillmore, Wooters, and Baker, 2001); and (5) and (6) are following PropBank corpus (Palmer, Gildea, and Kingsbury, 2005) annotation scheme. We will give a more detailed introduction of these two corpora in Section 1.2.1.

1.1.2 Semantic Role Labeling in NLP Applications

While predicate-argument structures might have various syntactic realizations, as shown in previous examples, semantic role labeling can offer a unified annotation for predicate-argument relation representation. This abstraction can facilitate many NLP applications.

In the following, we will describe how SRL relates to such applications as information extraction, document summarization, question answering, textual entailment (as mentioned in (Yih and Toutanova, 2006)), and some other NLP applications.

SRL for Information Extraction (IE)

The primary goal of the Information Extraction (IE) task is to provide those pieces of information that are salient to the user's needs. The kinds of information that IE systems extract vary in detail and reliability. For example, named entity recognition, entities relevant facts and attributes identification, and event-level indexing are all sub-tasks of IE. A novel IE paradigm is proposed by (Surdeanu et al., 2003) that takes advantage of predicate-argument structures where they built up a semantic role labeling system and used the extracted semantic role information to fill out the *template* slots for the purpose of IE. More recently, extracting meaningful relations among named entities from unstructured natural language text has attracted a lot of attention. Semantic role information is used as one of the important feature for relation extraction in (Harabagiu, Bejan, and Morarescu, 2005; Culotta and Sorensen, 2004; Bunescu and Mooney, 2005),

Predicate-argument structures (PAS) are also applied to IE tasks in the biomedical domain. Yakushiji et al. (2005) extracted the interaction relationships between proteins by taking advantage of PAS. Specifically, sentences are first passed to a full parser[1] and a PAS is extracted from the parses, which can absorb the diverse forms of surface sentences. This level of abstraction then facilitates automatic extraction of rules that are used to represent the interaction of proteins. Their results show that the performance is remarkably promising and comparable to manually-created extraction rules.

SRL for Question Answering (QA)

Current Question Answering (QA) systems extract answers from large text collections by (1) classifying the answer type they expect, (2) using question keywords or patterns associated with questions to identify candidate answer passages, and (3) ranking the candidate answers to decide which passage contains the exact answer. In (Narayanan and Harabagiu,

[1]Enju: `http://www-tsujii.is.s.u-tokyo.ac.jp/enju/`

2004), semantic role information is incorporated into both questions and documents; it first helps to identify the topic model that contributes to the interpretation of the questions, and then becomes particularly useful when building a scalable and expressive model of actions and events, which allows the sophisticated reasoning imposed by QA within complex scenarios. Here is an example with the corresponding semantic role labels in both the question and the answer to illustrate how this information might help:

Question	*What kind of nuclear materials were **stolen** from the Russian navy?*
srl(Q)	*What [$_{A1}$ kind of nuclear materials] were **stolen** [$_{A2}$ from the Russian Navy]?*
Answer	*Russia's Pacific Fleet has also fallen prey to nuclear theft; in 1/96, approximately 7 kg of HEU was reportedly stolen from a naval base in Sovetskaya Gavan.*
srl$_1$(A)	*[$_{A1}$ Russian's Pacific Fleet] has [$_{AM-DIS}$ also] **fallen** [$_{A1}$ prey to nuclear theft]; ...*
srl$_2$(A)	*... [$_{AM-TMP}$ in 1/96], [$_{AM-ADV}$ approximately] [$_{A1}$ 7 kg of HEU] was **stolen** [$_{A2}$ from a naval base] [$_{A3}$ in Sovetskaya Gavan].*
Result	exact answer = "approximately 7 kg of HEU"

Evaluation in (Narayanan and Harabagiu, 2004) shows that the percentage of questions whose types can be correctly identified increases from 12% to 32% (PropBank annotation) and to 19% (FrameNet annotation) respectively, due to integration of semantic role information.

SRL for Document Summarization

The DUC-2005 (*Document Understanding Conference 2005*)[2] competition task is to generate a 250-word summary based on the given questions and multiple relevant documents.

[2] `http://duc.nist.gov/`

Melli et al. (2005) integrated semantic role labeling component into their SQUASH system. In SQUASH, the ROUGE-2 score[3] on the development set increases from 0.0699 using Naive SRL to 0.0731 using ASSERT[4] SRL. This is a large improvement considering the impact of other successful features.

In the SQUASH system, semantic role information is used for *sentence selection* and *sentence compression*. For sentence selection, it contributes to estimating the significance score assigned to each sentence in terms of the semantic roles that the involved entities play in the sentence. For example, the following sentence is from Document 03 of question q0442g, DUC-2005. According to ASSERT output, it is labeled as:

> *The editor tried to console her by telling about* "[A0 *the guy in*] *the crowd*" [R-A0 *who*] "[V *saved*] [A1 *the President's life*]."

Based on their metric, this sentence is a good candidate for the given question "*What are some outstanding instances of heroic acts when the hero was in danger of losing his/her life while saving others in imminent danger of losing their own lives?*" (q0442g in DUC-2005).

In addition, it also helps to measure the similarity of sentences and remove the redundant information in the sentences to enable the summary to fit in the 250-word length constraint. For example, those constituents that are captured by the semantic role labels ARG-TMP (temporal markers) and ARG-DIS (discourse markers) are removed for sentence compression purposes.

SRL for Semantic Entailment

Semantic entailment is a task of determining, for example, if sentence (7) entails sentence (8).

---

[3]ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. It includes measures to automatically determine the quality of a summary by comparing it to other (ideal) summaries created by humans. The measures count the number of overlapping units such as N-gram, word sequences, and word pairs between the computer-generated summary to be evaluated and the ideal summaries created by humans. ROUGE-2 score is the measure for bi-grams in the summary. More details about ROUGE can be found in (Lin and Hovy, 2003).

[4]ASSERT stands for Automatic Statistical SEmantic Role Tagger, and can be downloaded from `http://www.cemantix.org/download/assert/beta/download.html`.

(7) *Walmart defended itself in court today against claims that its female employees were kept out of job in management because they are women.*

(8) *Walmart was sued for sexual discrimination.*

It is a fundamental problem in natural language understanding and is also the heart of many high level natural language processing tasks, including QA and IE.

In (Braz et al., 2005), semantic role information is extensively used in the hierarchical knowledge representation component and the inference component. The integration of SRL information greatly extends the richness of knowledge representation and therefore forms the basis of subsumption. Here is an example from (Braz et al., 2005). The task is to decide if S entails T.

S: *The Spanish leader razed Tenochtitlan in 1521 and constructed a Spanish city on its ruins.*

T: *The Spanish leader destroyed Tenochtitlan and built a Spanish city in its place.*

Their system identifies two verb frames in both S and T:

S-A: *[$_{A0}$ The Spanish leader] raze [$_{A1}$ Tenochtitlan].*

S-B: *[$_{A0}$ The Spanish leader] construct [$_{A1}$ a Spanish city] [$_{AM-LOC}$ on its ruins].*

T-A: *[$_{A0}$ The Spanish leader] destroy [$_{A1}$ Tenochtitlan]*

T-B: *[$_{A0}$ The Spanish leader] build [$_{A1}$ a Spanish city] [$_{AM-LOC}$ in its place].*

In this case, the lemmas of the key verbs in S and T will not exactly match. Since WordNet (Miller, 1995) contains synonym relations for "destroy" and "raze", "build" and "construct", the subsumption component determines that the verbs match. Consequently, by matching semantic arguments, the subsumption algorithm determines that, at the verb level, S entails T.

A conventional semantic entailment approach relies on mapping to a first order logic representation with a general theorem prover but without any rich knowledge source acquired from a particular domain. Compared with the first order logic representation, semantic role labeling provides semantic information which, while more general and domain-independent, is still adequate for this task. And it turns out that by combining SRL with the syntactic

level knowledge, the overall system performance increases from 54.7% to 65.9% on the RTE-PASCAL corpus[5].

SRL for Other NLP Applications

Machine Translation (MT) Wu and Fung (2009) proposed a two-pass statistical machine translation (SMT) system, where the first pass is performed using a conventional phrase-based SMT model, and the second pass is performed by a re-ordering strategy guided by semantic role labelers that produce both semantic frame and semantic role labels. The basic assumption is that "the semantic frames (target predicates and their associated semantic roles) should be all consistent between the input and output sentences, and are aligned to each other by the phrase alignments from the first pass". Otherwise, the second pass needs to re-order constituent phrases corresponding to predicates and arguments, seeking to maximize the cross-lingual match of the SRL output of the re-ordered translation to that of the original input sentence. Evaluation on a Wall Street Journal (WSJ) test set showed the hybrid model to yield an improvement of roughly half a point in BLEU score[6] over a strong pure phrase-based SMT baseline.

Verb Sense Disambiguation Dang and Palmer (2005) integrated PropBank features, such as labels of semantic roles and syntactic phrase type corresponding to each semantic roles, into their automatic Word Sense Disambiguation (WSD) system to improve the performance of verb sense disambiguation. The results show knowledge of gold-standard predicate-argument information from PropBank improves WSD on both coarse-grained senses and fine-grained WordNet senses. The positive effect of semantic role labeling on the verb sense disambiguation task is not a surprise, since the predicate-argument structures essentially encode the verb sense information. Verb sense disambiguation and semantic role labeling are in fact two closely related tasks.

Automatic Case Marker Prediction in Japanese Japanese case markers indicate the grammatical relation of the complement NP to the predicate. It often poses a challenge to Japanese text generation if performed by a foreign language learner, or by a machine

[5] `http://www.pascal-network.org/Challenges/RTE/`

[6] BLEU (Bilingual Evaluation Understudy) is an algorithm for evaluating the quality of text which has been machine-translated from one natural language to another. More details about BLEU can be found in (Papineni et al., 2002).

translation system. In (Suzuki and Toutanova, 2006), it is formulated as a semantic role labeling task in a monolingual setting, and the system outperforms the language model, based baseline systems significantly.

Coreference Resolution Ponzetto and Strube (2006) investigated the effects of using semantic role information in the coreference resolution system. Aside from the semantic knowledge extracted from WordNet, they also introduced (*semantic role argument*, *predicate*) pairs for both referring expressions (RE). Calibration of the features shows semantic role information helps to improve performance.

Noun Classification Hindle (1990) described an approach to classifying English words according to the predicate-argument structures they show in the corpus. It is based on the idea that for any noun, there is a restricted set of verbs that the noun appears as subject of or object of. For example, *wine* may be *drunk, produced*, and *sold* but not *pruned*. Each noun may therefore be characterized according to the verbs that it occurs with. Nouns may then be grouped according to the extent to which they appear in similar environments.

1.1.3 Semantic Role Labeling v.s. Semantic Parsing

Semantic parsing is the task of mapping a natural language (NL) sentence into a complete, formal, symbolic representation using a *meaning representation language* (MRL). It produces a deep semantic analysis which is a representation of the sentence in predicate logic or other formal language that supports automated reasoning; in contrast, the goal of semantic role labeling, as a form of shallow semantic parsing, is to identify and assign labels such as *agent*, *patient*, *manner* to the constituents in the sentence for a particular predicate, and need not generate *complete, formal meaning representations*.

Here is an example of MRL called CLANG, which is a team coaching language for the agents of RoboCup[7]:

– *((bpos (penalty-area our)) (do (player-except our {4}) (pos (half our))))*
– *"If the ball is in our penalty area, all our players except player 4 should stay in our half."*

[7] `www.robocup.org`

Largely due to the difficulty of developing an open-domain MRL and constructing a large annotated corpus of (NL, MR) pairs for domain general text, the evaluation of current semantic parsing systems (Ge and Mooney, 2005; Zettlemoyer and Collins, 2005; Ge and Mooney, 2006; Kate and Mooney, 2006; Wong and Mooney, 2006; Zettlemoyer and Collins, 2007) has been restricted to limited domains such as a simulated soccer domain (Mao et al., 2003) and a natural-language database interface of U.S. geography. Thus learning semantic parsers presents more challenges than semantic role labeling, and therefore has not been used as widely as semantic role labeling.

1.2 Learning an SRL system

1.2.1 FrameNet and PropBank

With the rapid development of machine learning techniques, annotated corpora are becoming more and more important in statistical NLP processing. In this section, we will give a description of the corpora used for semantic role labeling task.

The Berkeley FrameNet project (Baker and Fillmore, 1998) and PropBank (Palmer, Gildea, and Kingsbury, 2005) are two major annotation projects that produced text with semantic role annotation. However, there are some key difference between these two due to their different initial design philosophies and policy choices.

The following examples illustrate the different annotation schemes of FrameNet and PropBank for the same sentence.

FRAMENET ANNOTATION:

(9) *[$_{Buyer}$ Chuck]* ***bought*** *[$_{Goods}$ a car] [$_{Seller}$ from Jerry] [$_{Payment}$ for $1000].*

(10) *[$_{Seller}$ Jerry]* ***sold*** *[$_{Goods}$ a car] [$_{Buyer}$ to Chuck] [$_{Payment}$ for $1000].*

PROPBANK ANNOTATION:

(11) *[$_{A0}$ Chuck]* ***bought*** *[$_{A1}$ a car] [$_{A2}$ from Jerry] [$_{A3}$ for $1000].*

(12) *[$_{A0}$ Jerry]* ***sold*** *[$_{A1}$ a car] [$_{A2}$ to Chuck] [$_{A3}$ for $1000].*

In the following, we will introduce them from several aspects:

Goal: FrameNet is primarily a lexicographical project. Its starting point is the observation that words can be grouped into semantic classes, the so-called "frames", representations for prototypical situations or states. Each frame provides its set of semantic roles. The FrameNet project is building a dictionary which links frames to words and the expressions that can introduce them in text.

On the other hand, PropBank had the more practical aim of obtaining a complete semantic role annotation of the Penn Treebank. The PropBank lexicon was added first to facilitate annotation, and later evolved into a resource on its own. No higher-order organization was established at first, so for each unique verb sense, a "frameset" was constructed that consists of the set of semantic roles at its accompanying syntactic realizations.

Annotation Methodology:

One major difference in philosophy between FrameNet and PropBank is that FrameNet is primarily focused on the *type* of the verb frame (across many different tokens), while PropBank is primarily concerned with annotation of each *token* of a verb frame since they annotate every sentence in the Penn TreeBank.

The FrameNet project methodology has proceeded on a *frame-by-frame* basis, that is by first choosing a *semantic frame* (e.g., Commerce), defining the *frame* and its participants or *frame elements* (BUYER, GOODS, SELLER, PAYMENT), listing the various lexical predicates which invoke the *frame* (e.g., *buy, sell, etc.*) and then finding example sentences of each predicate in a corpus (BNC) and annotating each *frame element* in each sentence.

In contrast, PropBank defines semantic roles on a *verb-by-verb* basis. Specifically, for a particular predicate, a sample of sentences from the corpus containing this verb is examined and grouped into one or more major senses, and each major sense turns into a single frameset. Verb senses are considered distinct if they have distinct syntactic behavior, which correlates with different types of allowable arguments.

The semantic roles for each verb sense are numbered sequentially from Arg0 to Arg5. Arg0 is generally the argument exhibiting features of a prototypical Agent while Arg1 is a prototypical Patient or Theme. In addition to verb-specific numbered roles, a category of adjunct semantic roles is defined with tag ArgM and one of a set of "functional tags" (around 13 functional tags) denoting the role of the element within the predicate, such as ArgM-LOC (Locatives) and ArgM-TMP (Temporal markers). These functional tags can

also appear on numbered arguments.

Annotated Corpus: The primary corpus used by FrameNet is the British National Corpus (BNC) and its POS-tagged and lemmatised version. Parse trees are not used in FrameNet; the grammatical function tags are directly marked for the *frame elements* in the text. PropBank's annotation takes place with reference to the Penn Treebank trees: not only are annotators shown the trees when analyzing a sentence, they are constrained to assigning the semantic labels to portions of the sentence corresponding to nodes in the tree.

Released Versions: The major product of FrameNet project, the FrameNet lexical database, has gone through three releases. Release 1.3 (the latest one) currently contains more than 10,000 lexical units, over 6,000 of which are fully annotated, in nearly 800 hierarchically-related semantic frames, exemplified in more than 135,000 annotated sentences. Active research projects are seeking to produce comparable frame-semantic lexicons for other languages and to devise means of automatically labeling running text with semantic frame information, including German FrameNet (`http://gframenet.gmc.utexas.edu/`), Japanese FrameNet (`jfn.st.hc.keio.ac.jp/`) and Spanish FrameNet (`gemini.uab.es:9080/SFNsite`). In particular, a soccer FrameNet has been launched (`http://www.k-icktionary.de`) as a domain-specific trilingual (English, German and French) lexical resource of the language of soccer.

English PropBank I was released in Spring 2004 and it covers 3,323 predicate verbs and 4,659 framesets. Out of the 787 most frequent verbs, 521 have only 1 frameset and 169 have 2 framesets and 97 have 3 or more framesets. Note that framesets are not necessarily consistent between senses of the same verb; rather they are consistent between different verbs that share similar argument structures. Chinese PropBank (`http://www.cis.upenn.edu/~chinese/cpb/`) and Korean PropBank (`http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2006T03`) were released in 2005 and 2006 respectively. PropBank annotation has been explored in the biomedical domain (Chou et al., 2006) by adding the PropBank annotation on top of GENIA Treebank (GTB) (Tateisi et al., 2005).

Some SRL systems have been built based on FrameNet (Gildea and Jurafsky, 2002; Thompson, Levy, and Manning, 2003; Matsubayashi, Okazaki, and Tsujii, 2009), whereas

PropBank has been more widely used by the SRL research community for its close connection with Penn Treebank and its more generalized annotation for semantic role labels. In addition, PropBank provides more training examples (40K sentences more) than FrameNet, which makes it more appealing for SRL when applying machine learning method. In our implemented systems, we also use the PropBank corpus and focus on verbs as predicates.

NomBank

NomBank (`http://nlp.cs.nyu.edu/meyers/NomBank.html`) is an annotation project at New York University that is related to PropBank, which annotates nominal argument structure on the same corpus, using the same formalism as PropBank. Given nominalization/verb mappings, the combination of NomBank and PropBank affords even greater generalization. The goal is to annotate each "markable" NP, marking the head, its arguments, and "allowed" adjuncts in the style of PropBank. For example,

(13) students' knowledge of two-letter consonant sounds
ARG0 = students, REL = knowledge, ARG1 = two-letter consonant sounds

For more details of NomBank, refer to (Meyers et al., 2004). Jiang and Ng (2006) built a maximum-entropy based semantic role labeler on NomBank. CoNLL-2008 shared task (Surdeanu et al., 2008) also took into account the Nombank predicates.

1.2.2 Overview of a Semantic Role Labeling System

Researchers have taken many different computational approaches in analyzing semantic role labeling. Some traditional parsing and understanding systems (Pollard and Sag, 1994) rely more on hand-annotated grammars, which are typically time-consuming to create and often have limited coverage. Data-driven techniques that were applied to template-based semantic interpretation in limited domains often perform only shallow syntactic analysis (Miller et al., 1996; Riloff, 1993; Riloff and Schmelzenbach, 1998). More recently, with the availability of annotated corpora like PropBank and FrameNet and a variety of machine learning techniques, there has been rapid development of research into corpus-based SRL approches (Gildea and Jurafsky, 2002; Gildea and Palmer, 2002; Surdeanu et al., 2003; Chen and Rambow, 2003; Gildea and Hockenmaier, 2003; Xue and Palmer, 2004; Pradhan

et al., 2004; Pradhan et al., 2005). The shared task in Senseval-3 (`http://www.senseval.org/senseval3`), CoNLL-2004 (Carreras and Màrquez, 2004) and CoNLL-2005 (Carreras and Màrquez, 2005) was on SRL; and the shared task in CoNLL-2008 (Surdeanu et al., 2008) and CoNLL-2009 (`http://ufal.mff.cuni.cz/conll2009-st/`) was on joint parsing of syntactic and semantic dependencies, which is also relevant to SRL. In this section, we will give an overview of the current main-stream SRL systems.

Development of SRL Systems

Traditional parsing and understanding systems, including implementations of unification-based grammars such as HPSG (Pollard and Sag, 1994), rely on hand-developed grammars, which must anticipate every way in which semantic roles may be realized syntactically. Writing such grammars is time-consuming, and typically such systems have limited coverage.

Data-driven techniques have been used for template-based semantic interpretation in limited domains to avoid complex feature structures, and often perform only shallow syntactic analysis. For example, in the context of the Air Traveler Information System (ATIS) for spoken dialogue, Miller et al. (1996) computed the probability that a constituent such as *Atlanta* filled a semantic slot such as DESTINATION in a semantic frame for air travel. In a data-driven approach to information extraction, Riloff (1993) builds a dictionary of patterns for filling slots in a specific domain, such as terrorist attacks, and (Riloff and Schmelzenbach, 1998) extends this technique to automatically derive entire "case frames" for words in the domain. They make use of a limited amount of hand labour to accept or reject automatically generated hypotheses. They show promise for a more sophisticated approach to generalizing beyond the relatively small number of frames considered in the tasks. Don Blaheta and Eugene Charniak (2000) presented a domain-independent system of assigning function tags such as MANNER and TEMPORAL included in the Penn Treebank corpus. Some of these tags correspond to the semantic roles defined in FrameNet and PropBank, but the Treebank tags do not include all the arguments of most predicates.

More recent work in developing SRL systems aims for a statistical system to learn to identify and classify all the semantic roles for a wide variety of predicates in unrestricted text. Gildea and Jurafsky (2002) presented the first statistical SRL system on FrameNet.

This system is based on a statistical classifier trained on roughly 50,000 sentences that are extracted from FrameNet corpus. Each training example is parsed into a syntactic tree using the Collins' Parser (Collins, 1997) and a set of syntactic and lexical features, such as the *phrase type* of each constituent, its *position*, etc., are extracted. These features are combined with knowledge of the predicates, as well as information such as the prior probabilities of various combinations of semantic roles. This work lays the foundation for the current automatic semantic role labeling systems. It constructs a general SRL system architecture and provides a core feature set that has been widely used by almost all the current SRL systems.

Senseval-3 and CoNLL shared tasks To further the accessibility of FrameNet and its amenability to NLP applications, Senseval-3 called for the development of systems to meet the same objectives as the study in (Gildea and Jurafsky, 2002). The basic task for Senseval-3 is: given a sentence, a target word and its frame, identify the frame elements within that sentence and tag them with the appropriate frame element name. The Senseval-3 task uses approximately 8,000 randomly selected sentences and 40 randomly selected frames. Evaluation of the system follows the metrics[8] of the study in (Gildea and Jurafsky, 2002). Most concentrated work of SRL on FrameNet is stimulated by this task and achieves overall better results in comparison with (Gildea and Jurafsky, 2002)'s study. A summary of the systems can be found in (Litkowski, 2004).

The shared tasks of CoNLL-2004 and CoNLL-2005 were on the semantic role labeling task based on PropBank predicate-argument structures. Given a sentence, for each target verb in the sentence, all the constituents that fill a semantic role of the verb have to be recognized. In CoNLL-2004, the goal was to develop SRL systems based on partial parsing information; in CoNLL-2005, the focus was to increase the amount of syntactic and semantic input information in order to boost the performance of machine learning systems on the SRL task. Due to the availability of full parsing information and the application of advanced learning strategies, the systems competing in CoNLL-2005 showed a great improvement over those in the previous year. In particular, this competition produced

[8] *Precision, Recall, Overlap* and *Attempted*. *Overlap* was the average overlap of all correct answers. The percentage *Attempt* was the number of frame elements generated divided by the number of frame elements in the test set

many state-of-the-art SRL systems, and some of these systems have been released and widely used for research[9]. For more detailed descriptions about the task, data, systems and evaluations see `http://www.lsi.upc.edu/~srlconll/`.

The goal of the shared task of CoNLL-2008 (`http://barcelona.research.yahoo.net/conll2008/`) is to explore the possibility of modeling parsing and semantic role labeling in the same dependency-based framework. Besides verbal predicates, it also considers nominal predicates. Compared to the SRL task defined in CoNLL-2004 and CoNLL-2005, this task also involves the parsing of syntactic dependencies. The evaluation measures for this task are multi-dimensional. The evaluation is *root-based*, where a case is counted as positive as long as the root of the argument is correctly identified. It evaluates not only the dependency between the predicate and each of its argument, but also the predicted sense of the current predicate. From the submitted systems, it was shown that the extraction of syntactic and semantic dependencies can be performed with state-of-the-art performance with a pipeline of linear complexity components (Ciaramita et al., 2008), which makes this technology attractive for the real-world applications. The CoNLL-2008 shared task was defined on English only. CoNLL-2009 extended the same task to multi-lingual setting.

These competitions greatly promoted the development of SRL systems and produce benchmarks for data sets and systems evaluations which have been used in later research. Since our focus is on building an SRL system based on given syntactic structures, we will be more interested in a setting such as that given in CoNLL-2005. We continue our introduction in this context.

General Architecture of an SRL System

Figure 1.1 shows the overview of a discriminative SRL system. For a given sentence and target predicates in the sentence, an SRL system has to identify all the constituents in the sentence that can fill some semantic role for each predicate. The Annotator component enriches the sentence using syntactic parsers (syntactic parse trees) or shallow parsers (chunking information) or semantic ontologies such as WordNet/VerbNet, or Named Entity Identifier (Named Entity information). Then the Feature Extractor component extracts pre-defined features such as the phrase type of the constituents, voice of the predicate,

[9] such as **ASSERT:** `http://www.cemantix.org/download/assert/beta/download.html`

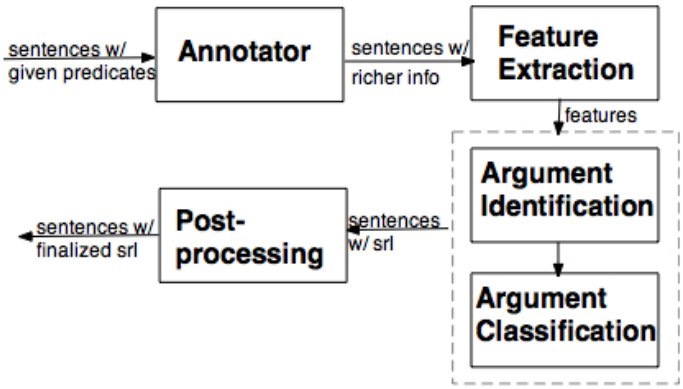

Figure 1.1: Overview of a semantic role labeling system

predicate lemma, etc. from the output of the Annotator.

The extracted features are fed into the Learning component to train a classifier. The Learning component is typically divided into two sub-components: *the semantic role identification component* and *the semantic role classification component*. Semantic role identification is the process of identifying parsed constituents in the sentence that represent semantic arguments of a given predicate; semantic role classification is the process of assigning the appropriate argument labels to a given constituent that has known to represent some argument of a predicate. Assuming the Annotator provides a syntactic tree to the sentence, then each node in the parse tree can be classified either as a semantic argument (i.e., a NON-NULL node) or one that does not represent a semantic argument (i.e., a NULL node). The purpose of semantic role identification is to distinguish the NULL nodes from the NON-NULL nodes. The NON-NULL nodes may then be further classified with the set of argument labels by the semantic role classification component. The rationale of splitting the task into identification and classification is twofold.

1. Splitting the task can increase computation efficiency in training. In identification phase, every parse tree constituent is a candidate for the classifier; therefore each parse tree contains 40 candidates on average and we have around 40,000 sentences in the training set. If we only deal with the NON-NULL nodes in the classification phase, then only a small number of candidates are involved in the computation, 2.7

constituents on average per parse tree (for each predicate).

2. Different features might be helpful for each task. For example, structural features are more helpful for identification while lexical features are more helpful for classification (Pradhan, Ward, and Martin, 2008).

The learning component uses a local model, where the semantic role probability estimation for each constituent is independent of one another. Under this strong independence assumption, systems can achieve acceptable performance. Some hard constraints, such as no overlapping between constituents that are labeled with semantic roles or the sequence of semantic role labels has to be "legal", were imposed as a post-processing step. Some systems take into account the interdependence among the labels of constituents in a "global/joint" learning framework and achieve better performance at the cost of a more complicated learning strategy. We will talk more about this in Section 1.3.3.

A Typical Setting for SRL

In this section, we will describe a typical setting for a discriminative semantic role labeling system.

Annotation used: For a given sentence, the syntactic parse tree of the sentence is generally taken as the input of an SRL system and each substring of the sentence that has a semantic label typically corresponds to a syntactic constituent in the tree. Identifying constituents is especially important for the PropBank since the annotation process involved augmenting the constituents in the Penn Treebank with semantic role information.

To date, the most commonly used annotation used for SRL task is the full syntactic parse tree on the PropBank predicate-argument annotations. There have also been some attempts at relaxing the necessity of using full syntactic parse trees to using dependency structures or shallow syntactic information at the level of phrase chunks. Based on the syntactic annotations of the input sentence described above, features are extracted from the syntactic parse trees. A particular set of such features has been widely used in SRL systems. We will give more details of features that used in our system in later chapters.

Learning component: A variety of learning strategies have been explored, such as generative Bayesian probabilistic models (Gildea and Jurafsky, 2002; Gildea and Palmer,

2002; Gildea and Hockenmaier, 2003), Decision Trees (Chen and Rambow, 2003), Maximum Entropy (Xue and Palmer, 2004), Support Vector Machines (SVMs) (Pradhan et al., 2004), Tree Conditional Random Fields (Cohn and Blunsom, 2005). Thompson et al. (2003) also explored a generative model for FrameNet SRL system. SVMs are one of the most widely used methods in the community.

Evaluation In general, the standard measures for the performance of SRL systems are *Precision, Recall and F-score.* For each semantic role, such as A0, the *Precision, Recall and F-score* are calculated and overall system performance is evaluated in terms of the number of correctly labeled arguments, the number of labeled arguments and the number of gold arguments[10]. Aside from the overall system evaluations, evaluations for subtasks of argument label identification and classification are also given to obtain a better comparison and analysis to the systems. Precision, recall and f-score are calculated as follows:

Precision = number of correctly labeled arguments / number of labeled arguments
Recall = number of correctly labeled arguments / number of gold arguments
*F-score = 2 * precision * recall / (precision + recall)*

In some case when the "NULL" arguments are also taken into account or the number of labeled argument equals to the number of gold arguments, another measure is used:

Accuracy = number of correctly labeled constituents/ number of target constituents

Example:

Gold [$_{A0}$ The girl] **broke** [$_{A1}$ the mirror] [$_{AM-TMP}$ yesterday].
Guess [$_{A0}$ The girl] **broke** the [$_{A1}$ mirror] [$_{AM-LOC}$ yesterday].

Gold	Guess	
The girl → A0	The girl → A0	correct boundary, correct label
The mirror → A1	mirror → A1	wrong boundary, –
yesterday → AM-TMP	yesterday → AM-LOC	correct boundary, wrong label

[10] in this case, only positive examples are considered

Overall Evaluations:

Precision, Recall, F-score: 1/3, 1/3, 1/3

Evaluations on Subtasks:

Identification(*Precision, Recall, F-score*): 2/3, 2/3, 2/3
Classification(*Accuracy*): 0.5

Some State-of-the-Art SRL Systems

Researchers have been tackling different aspects of this task, and due to the variety of knowledge resources, data sets and evaluations used in these systems, performance varies accordingly. The CoNLL-2005 shared task provides a common platform to showcase some state-of-the-art SRL systems. Here we will give a brief description of the top two SRL systems in this competition.

The top system (Punyakanok, Roth, and Yih, 2005a) achieves (p/r/f%) 82.82/76.78/79.44 overall performance on test data provided by the shared task. It takes the output of multiple argument classifiers and combines them into a coherent predicate-argument output by solving an optimization problem. The optimization stage, which is solved via integer linear programming, takes into account both the recommendation of the classifiers and a set of linguistic and structural constraints that the global argument label assignment has to be subject to, and is thus used both to clean the classification results and to ensure structural integrity of the final role labeling. The system shows some improvement through this inference step. After this, a better result (p/r/f%) 81.90/78.81/80.32 has been reported in (Toutanova, Haghighi, and Manning, 2008). To our knowledge, so far it is the best reported performance on CoNLL-2005 shared task. Instead of using the single given Charniak parse (from Charniak parser version 2000) in CoNLL data, they used the top-10 automatic parse trees from the May 2005 version of the Charniak parser (Charniak and Johnson, 2005) with correction of forward quotes[11].

[11]The Charniak parses provided as part of the CoNLL shared task data uniformly ignore the distinction between forward and backward quotes and all quotes are backward. In (Toutanova, Haghighi, and Manning, 2008), they re-ran the parser and obtained analyses with correct treatment of quotes, which leads to over 1% improvement in f-score in their joint SRL model.

The 2nd system (Pradhan et al., 2005c) achieves (p/r/f%)82.95/74.78/78.63 overall performance on the same data set. In this system, for an input sentence, syntactic constituent structure parses are generated by Charniak's parser and Collins' parser. Semantic role labels are assigned to the constituents of each parse using SVM classifiers. The (two) resulting semantic role assignment sequences are then converted to an IOB representation, which are used as additional features, along with the flat syntactic chunks, by a chunking SVM classifier that produces that final SRL output. This strategy for combining features from three different syntactic views gives a significant improvement in performance over roles produced by using any one of the syntactic views individually. This system has been released and widely used for research[12]. In (Pradhan, Ward, and Martin, 2008), the robustness of this system is carefully examined and analyzed when trained on one genre of data and used to label a different genre. Their experiments are based on comparisons of performance using PropBank-ed WSJ data and PropBank-ed Brown Corpus data. The results indicate that "whereas syntactic parses and argument identification transfer relatively well to a new corpus, argument classification does not" (Pradhan, Ward, and Martin, 2008).

The common ground underlying these two systems is that they combine the outputs from different syntactic views. When an SRL system takes syntactic parse trees as input, the incorrect parses would become one of the major error sources because the syntactic parser may fail to produce a constituent that corresponds to the portion for the correct semantic argument. The hope of combining different syntactic views is that they may give different errors, and combining them will be better than either system alone. Punyakanok et al. (2005a) deal with the multiple outputs by formalizing it into an optimization problem and imposing the hard constraints on the predicted sequence. Pradhan et al. (2005c) add the IOB features to the phase-based chunker in order to take advantage of the accuracy gained from the full parses while preserving the robustness and flexibility of the chunker. The top system focuses more on producing the globally legitimate sequence and the 2nd-ranked system aims more at fixing the errors caused by the inaccurate parser.

[12] `http://www.cemantix.org/download/assert/beta/download.html`

1.3 Research Directions in Semantic Role Labeling

In the previous section, we gave an overview of a few points of interest involving SRL systems, including task description, NLP applications involving tasks, a typical setting of the task, and the development of SRL systems and state-of-the-art SRL systems. In this section, we will take a closer look at the task. The issues we are discussing in this section have been attracting a lot of interest from the community and some of them are still open to solve.

In current state-of-the-art SRL systems, syntactic information is used extensively as the main source of feature extraction; and the system performance relies heavily on the performance of the syntactic parsers. Based on the situation, currently there are 4 issues that are of interest to the SRL researchers: (i) exploring novel sources for predictive SRL features extraction, (ii) increasing the robustness of the SRL system, (iii) capturing predicate frames using joint models and global inference, and (iv) taking advantage of additional knowledge sources.

1.3.1 Development of Predictive Syntactic Features

The development of a proper set of features has been viewed as a crucial component in many machine-learning related applications. SRL systems generally take as input a syntactic parse tree and use the syntactic information as features to tag the semantic labels on the syntactic constituents. Based on this setting, Gildea and Jurafsky (2002) proposed a set of features in a statistical SRL system on FrameNet which has been taken as a set of baseline features in most of the current SRL systems[13]). Gildea and Palmer (2002) adopted the same feature set and the same statistical method on PropBank corpus. Surdeanu et al. (2003) and Pradhan et al. (2004) explored the parse tree further and proposed an additional set of features, among which are the generalizations of the more specific features such as named entities, POS of the head word and verb clusters. A series of analytic experiments show the contributions of each of the features/feature combinations to the performance improvement. Xue and Palmer (2004) take a critical look at the features used in the previous SRL systems and shows that the syntactic parse tree as the main

[13]We list all the features mentioned in this section in Table 3.1 of Chapter 3.

feature source has yet to be fully exploited. They propose an additional set of features, *syntactic frame* feature and some feature combinations in particular, and experimentally show that different features are needed for different subtasks; and the results with fewer features if properly applied, are comparable to the best previously reported results with a full set of features. Their work indicates that developing features that can capture the right kind of information is crucial to advancing the state-of-the-art SRL systems.

Aside from the full syntactic parses, features extracted from shallow parses are also explored (Hacioglu et al., 2004). In order to generalize the *path* feature which is probably the most salient (while being the most data sparse) feature for SRL, research efforts have been made to extract features from other syntactic representations, such as CCG derivations (Gildea and Hockenmaier, 2003) and dependency trees (Hacioglu, 2004) or integrate features from different syntactic parsers (Pradhan et al., 2005). Chen and Rambow (2003) and Liu and Sarkar (2006) also explored features from Tree-Adjoining Grammar (TAG) based notation. While those works exploit novel sources for SRL feature extraction from different syntactic views, they still rely on the constituency parses to some degree. Either the syntactic structures they used (such as dependency trees in (Hacioglu, 2004)) are converted from constituency parses, or the "novel features" from these syntactic structures are added to the "standard feature" set extracted from the constituency parses to form the new feature set applied to the system.

To avoid explicitly developing predictive syntactic features on trees, Moschitti (2004) used convolution kernels on selective portions of syntactic trees. It saves the effort of explicit feature selection and typically gains a moderate performance in contrast to the feature selection based SRL systems.

Researchers believe that there is still room for SRL performance improvement with a better feature selection strategy.

1.3.2 Robustness of SRL systems

There's an increasing amount of attention being paid to the robustness of SRL systems to parser errors, changing domains, unseen predicates, etc.

Robustness to Parser Errors

There are two sources of errors for semantic role identification: (i) failures by the system to identify all and only those constituents that correspond to semantic roles, when those constituents are present in the syntactic analysis, and (ii) failures by the syntactic analyzers to provide the constituents that align with correct arguments. Adding new features can improve performance when the syntactic representation being used for classification contains the correct constituents. Additional features can't recover from the situation where the parse tree being used for classification doesn't contain the correct constituent representing an argument. Increasing the robustness of SRL to the parser errors is essentially dealing with the second type of errors.

In practice, an SRL system experiences a dramatic drop in performance when working on automatic parses; the gap is typically around 10% (in f-score) (Toutanova, Haghighi, and Manning, 2005; Punyakanok, Roth, and Yih, 2005b; Xue and Palmer, 2004). To bridge the gap, some research efforts have been made which fall into two categories: (i) different syntactic views as complementary to each other, (ii) integration of parsing and semantic role labeling.

The basic idea of applying alternative syntactic views lies in the fact that different syntactic representations may contain different errors. The hope is that the combination of these different views could produce better results than any single one. In (Pradhan et al., 2005), three SRL systems are trained on three different syntactic views from Charniak's parser (Charniak, 2000), Lin's Minipar (Lin and Pantel, 2001) and the chunking parser (Hacioglu, 2004), respectively; then a voting strategy is applied when these SRLs run on test data. This strategy gives 3.8% improvement in f-score for argument identification and 2.2% improvement in f-score on the full task. Similarly, Punyakanok et al. (2005a) use Collins' parser's output and Charniak's parser's top-5 outputs as the sources of argument prediction. Then an optimization strategy is applied to obtain a global optimal argument sequence. This system was the best system submitted to the CoNLL 2005 shared task evaluation.

Yi and Palmer (2005) and Sutton and McCallum (2005) reported on the first attempt to integrate parsing and semantic role labeling to improve SRL performance. In (Yi and Palmer, 2005) new parsers are trained on the trees which are augmented with semantic

role labels and new parses turn out to have bigger coverage of the correct constituents; Sutton and McCallum (2005) use an SRL system to re-rank the top-N parses from the Charniak parser and try to pick the parse that is best for the SRL task. Sutton and McCallum (2005) do not try to augment the search space of the statistical parser based on the SRL task in contrast with (Yi and Palmer, 2005). Unfortunately, neither of them produces positive results: the new parser's output in (Yi and Palmer, 2005) performs worse than using Charniak's parser directly and top-1 parse always stands out in (Sutton and McCallum, 2005). It appears that when integrating parsing and semantic role labeling, somehow both performances deteriorated.

In (Pradhan, Ward, and Martin, 2008), the robustness of their system is carefully examined and analyzed when trained on one genre of data and used to label a different genre. It was shown that "errors in the syntactic parse are not a large factor in the overall performance difference (when the system is run on a different genre of data). Even more telling, there is still a large drop in performance when training and testing using Treebank parses".

Work on Semantic Role Annotations

Even though PropBank is the most widely used corpus for training SRL systems, a serious criticism to this corpus refers to the role set it uses, which consists of the set of numbered core arguments, whose semantic translation is verb-dependent. While Arg0 and Arg1 are intended to indicate the general roles of Agent and Theme, other argument numbers do not generalize across verbs and do not correspond to general semantic roles. Based on the hypothesis that a set of less verb-dependent semantic roles should be easier to learn and port better to different genres, a new set of semantic roles was developed by transforming PropBank roles to less verb-dependent thematic roles based on the mapping between PropBank and VerbNet (Loper, Yi, and Palmer, 2007). Yi et al. (2007) have shown that the SRL system that was trained on the new set of semantic roles confirms the hypothesis and gains the improvement of the overall results (f-score) on the WSJ test set by 6% and on the Brown corpus by almost 10%. An important remark to this work is that the improvement is mainly from Arg2 for which the new role is a combination of grouped VerbNet roles and for the rest of arguments, PropBank roles are still being

used. In contrast, Zapirain et al. (2008) presented their result by empirically testing a state-of-the-art SRL system with the two alternative role sets of PropBank and VerbNet respectively and concluded that the PropBank role set is more robust to the lack of verb-specific semantic information and generalizes better to the unseen predicates. How to improve the annotation of an SRL corpus to further improve the robustness of SRL systems remains an interesting point of research.

1.3.3 Joint Models and Global Inference

Despite the recent progress on accurate semantic role labeling, most of the work has largely used independent local classifiers which assign a label to an individual parse tree node without knowing the labels of other nodes. This obviously ignores the linguistic observation that a core argument frame is a *joint* structure, with strong dependencies between arguments. Some SRL systems have incorporated such dependencies in different ways. For example, Gildea and Jurafsky (2002) and Pradhan et al. (2004) encode it into *hard* constraints – that arguments cannot overlap with each other or the predicate, and also *soft* constraints – it is unlikely that a predicate will have two or more AGENT arguments, or a predicate used in an active voice will have a THEME argument prior to an AGENT argument. Punyakanok et al. (2008) use a global inference strategy subject to these constraints on the multiple SRL outputs. Toutanova et al. (2008) shows greater gains can be obtained by modeling the joint information about a predicate's argument structure. In their system, top-k non-overlapping assignments are selected first from the local model using dynamic programming and the best one is selected from these top-k in a joint log-linear model in a discriminative re-ranking setting. The features used in the joint model are more "predicate argument frame" related features such as *whole label sequence* (excluding modifying arguments).

Similar to (Toutanova, Haghighi, and Manning, 2008), (Johansson and Nugues, 2008a) presented an SRL system that is integrated with a dependency parser. The syntactic subcomponent is a pseudo-projective dependency parser and the semantic model uses a global inference mechanism on top of a pipeline of classifiers which takes as input the output of the dependency parser. The complete syntactic-semantic output is selected from a candidate pool generated by the subsystems by applying a syntactic-semantic reranker

to the top-16 trees from the syntactic module. There are only 3 features they considered for reranking: the log probability of the syntactic tree, the log probability of the semantic structure according to the pipeline, and the global model, respectively. This system is reported to achieve a near state-of-the-art performance when evaluated on CoNLL-2005 data.

The shared tasks of CoNLL-2008 and CoNLL-2009 also aimed to explore the possibility of jointly modeling parsing of syntactic and semantic dependencies in order to improve the performance of both. It turns out only 5 systems out of 55 actually combined the syntactic and semantic tasks in their implementation – all others still followed a pipelined architecture for this joint modeling task. In fact, for computational reasons, it is hard to encode the interdependency among (core) semantic argument into the classifying process. It is either encoded into the features or into constraints as post-processing.

1.3.4 Integration of Multiple Knowledge Sources

Although both FrameNet and PropBank are useful resources for semantic role labeling and are built with extensive human effort over years of work, most of the current study has been relying on only one of them, mainly due to the lack of connectivity between these resources that would enable their exploration in a unified way.

FrameNet is built based on semantic frames and therefore it provides a good generalization across predicates using frames and semantic roles. It also provides empirical evidence for the syntactic realization of these semantic frames. Despite the huge human efforts of constructing this lexical resource, its verb coverage is still very limited (3040 verbs attached to 321 distinct frames) and only 30% of these frames were considered to have enough training data and were used in the Senseval-3 task. Therefore broader frame coverage and more training data are necessary for a greater utility of FrameNet.

In contrast, PropBank was built on per-predicate basis and formed by a verb lexicon (Frameset Files) and a semantically annotated corpus. The lexicon contains about 3600 verbs which corresponds to 5000+ framesets. Most verbs (80%) in PropBank have one frameset, which generalizes over different senses of the verb. One of the main problems

of PropBank is that it is too domain-specific (finance domain). Also, due to its verb-by-verb based construction methodology as well as the shallow annotation of predicate-argument structures (For example, Arg2-4 are seriously "overloaded" due to the coarse-grained verb sense disambiguation), barriers to improving the generalization of current (PropBank-based) SRL systems across genres and towards deep semantic analysis still exist.

Two other useful lexicon resources, VerbNet (Kipper, H.T.Dang, and Palmer, 2000) and WordNet (Miller, 1995) are useful when linking PropBank and FrameNet. VerbNet is a verb lexicon with explicitly stated syntactic and semantic information based on Levin's verb classification (Levin, 1993). The fundamental assumption is that the syntactic frames of a verb as argument-taking elements are a direct reflection of the underlying semantics. Verb entries in the same VerbNet class share common syntactic frames, and thus they are believed to have the same syntactic behavior – this property can be used to extend the coverage of FrameNet: by identifying the VerbNet class that corresponds to a FrameNet frame, the verbs that are not covered by FrameNet can be included now. WordNet covers approximately 150,000 words which are organized in *synset*[14] and semantic links among these synsets. WordNet covers a large number of (around 11,000 verbs that are divided into 24,632 senses) but lacks information about verb syntax, which is of primary importance when creating a complete resource for working on verb behaviors.

These resources provide different information, thus mapping and integrating syntactic-semantic information among them is becoming a main concern today. In (Kipper, Palmer, and Rambow, 2002; Kipper, Snyder, and Palmer, 2004), a mapping between VerbNet and PropBank is proposed. In (Kipper, Snyder, and Palmer, 2004) each PropBank frameset has been manually linked, when possible, to the VerbNet class; each PropBank role is then mapped to the corresponding VerbNet thematic roles[15]. Shi and Mihalcea (2005) propose a mapping between VerbNet classes and FrameNet frames to build a unified resource for semantic parsing. In (Giuglea and Moschitti, 2006) the link between VerbNet and PropBank proposed in (Kipper, Palmer, and Rambow, 2002) is used together with a

[14]Each synset represents a lexicalized concept that is linguistically represented by a set of synonymous words and a gloss describing the synset itself.

[15]78.62% of PropBank sentences have an exact matching to a VerbNet class.

semi-automatic mapping from VerbNet to FrameNet to improve the performance of an SRL system. Pazienza et al. (2006) propose a mapping to link WordNet 2.0 to VerbNet 2.0 and finally to the PropBank corpus to obtain a large set of linguistic examples of verb pairs that have some semantic and specific predicate-argument structures. It provides a possibility to study and automatically learn how the predicate-argument structures of two verbs are related using the set of corpus examples. Currently there is a unified (FrameNet, VerbNet and PropBank) verb lexicon available at `http://www.cs.rochester.edu/~gildea/Verbs/`. How to apply this richer information to the SRL systems is not yet a well-studied issue.

1.4 Lexicalized Tree-Adjoining Grammars (LTAGs)

Our main focus in this thesis is to explore the utility of the Lexicalized Tree-adjoining grammars (LTAGs) formalism for the SRL task. In this section we provide our motivation for this approach, and an introduction to LTAG.

Tree Adjoining Grammar (TAG) (Joshi, Levy, and Takahashi, 1975) is a formal tree rewriting system whose formal properties have been extensively studied in recent years. It has been applied in some of the NLP applications such as machine translation (Abeillé, Schabes, and Joshi, 1990; Deneefe and Knight, 2009) and language generation (Stone and Doran, 1997; Harbusch and Woch, 2002). TAGs are often described as *mildly context-sensitive*, meaning that they possess certain properties that make them more powerful (in terms of the weak generative capacity) than context-free grammars (CFGs), but less powerful than context-sensitive grammars. Mildly context-sensitive grammars are conjectured to be powerful enough to model natural languages while remaining efficiently parseable in the general case (Joshi, 1985). LTAG stands for lexicalized TAG. Since the two are formally equivalent, they are often used interchangeably. In this section, we will give an introduction to the basics of LTAG. A recent review can be found in (Joshi and Schabes, 1997; Abeillé and Rambow, 2001), which provides a detailed description of TAG with respect to linguistics, formal and computational properties.

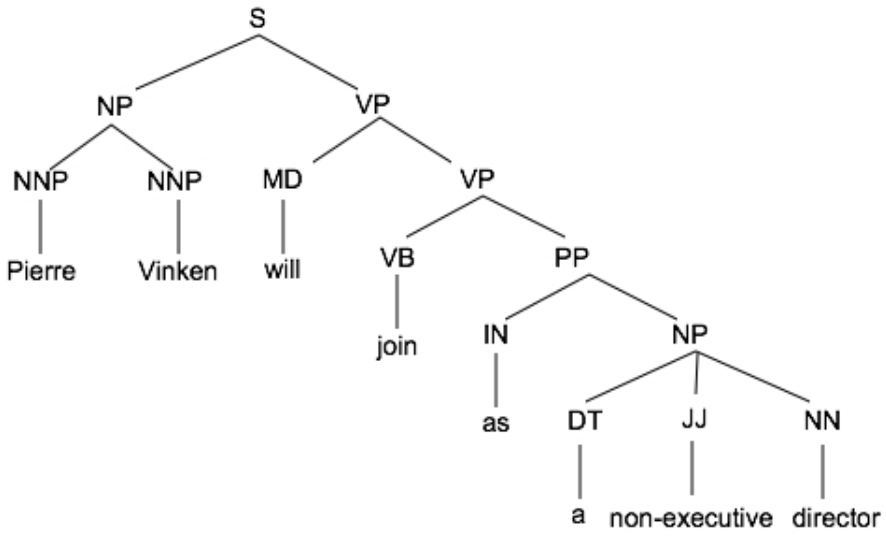

Figure 1.2: A parse tree

1.4.1 Elementary Trees

The primitive elements of an LTAG are *elementary trees* (etree for short). An LTAG is *lexicalized*, as each elementary tree is associated with one lexical item which is called the *anchor* of the tree. This lexical item is located on the frontier of the tree. Although the LTAG formalism allows wide latitude in how elementary trees may be defined, various linguistic principles generally guide their formation. An important principle is that dependencies, including long-distance dependencies are typically localized in the same elementary tree by appropriate grouping of syntactically or semantically related elements. There are two types of elementary trees: *initial trees* and *auxiliary trees*. Each auxiliary tree has a unique leaf node, called the *foot* node, which has the same label as the root and is usually marked with $\star$. In both types of trees, leaf nodes other than anchors and foot nodes are called *substitution* nodes which are usually marked with a down arrow. We illustrate the LTAG trees with an example.

(14) Pierre Vinken will join as a non-executive director.

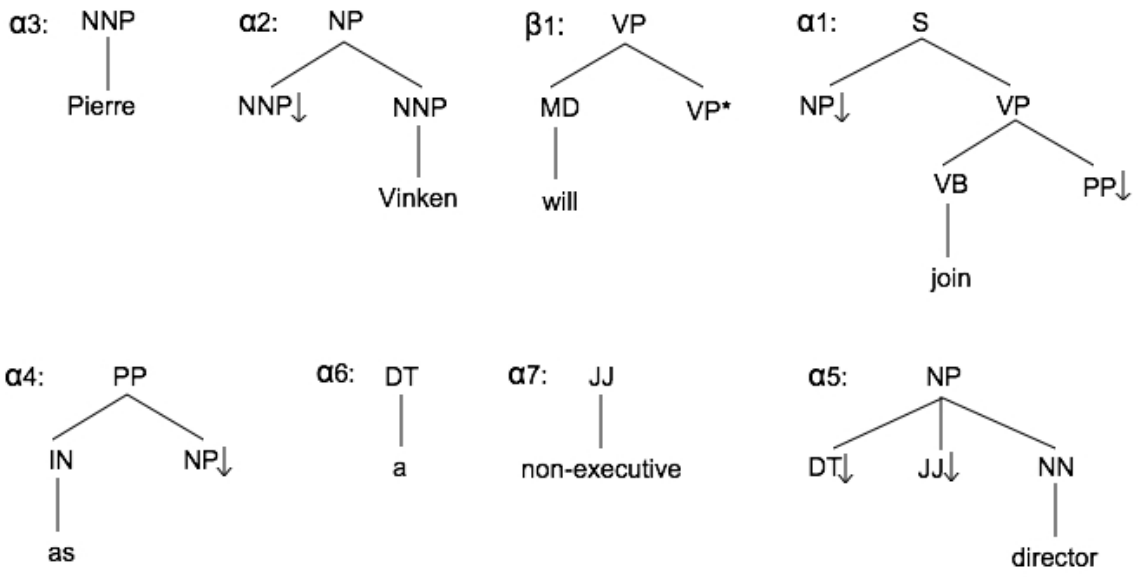

Figure 1.3: A set of elementary trees

The parse tree for the example is shown in Figure 1.2. Figure 1.3 shows the elementary trees for each word in the sentence. α stands for an initial tree, and β stands for an auxiliary tree. Note that LTAG is a formal grammar and not a linguistic theory, so elementary trees can be defined in many ways. Figure 1.3 just shows one of the many possible sets of etrees.

1.4.2 Two Operations

Typically etrees can be combined by two operations: *substitution* and *adjunction*. In the substitution operation, a substitution node in an etree is replaced by another etree whose root has the same label as the substitution node. In an adjunction operation, an auxiliary tree is inserted into another etree. The root and the foot nodes of the auxiliary tree must match the node label at which the auxiliary tree adjoins. Figure 1.4 and Figure 1.5 show the two operations. Figure 1.6 shows the LTAG derivation tree $\gamma 1$ generated by applying the two operations to the elementary trees in Figure 1.3. The structure that is the result

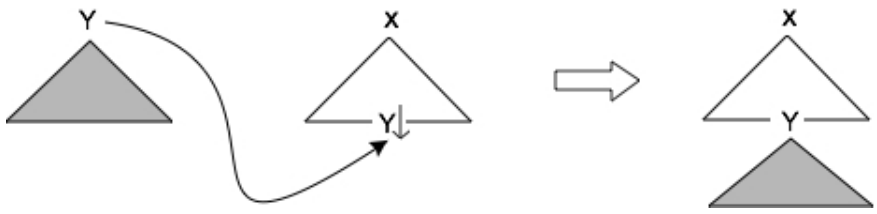

Figure 1.4: The substitution operation

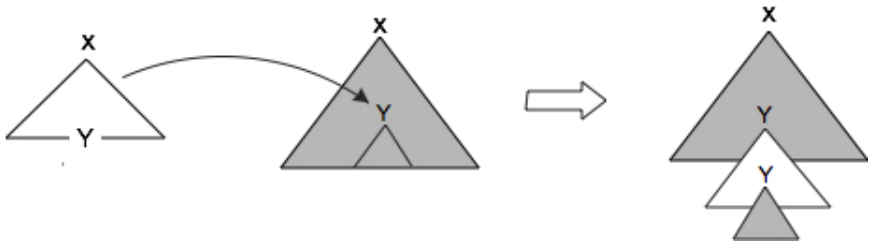

Figure 1.5: The adjunction operation

of combining together the elementary trees in an LTAG derivation is called a *parse tree* or a *derived tree*. The history of the combination process is recorded as a *derivation tree*.

1.4.3 Derived Trees and Derivation Trees

For semantic computation the LTAG derivation tree is the crucial object. Compositional semantics is defined on the derivation tree. The idea is that for each elementary tree there is a semantic representation associated with it and these representations are composed using the derivation tree.

Unlike in CFGs, the derived trees and the derivation trees in LTAG formalism are not identical, as previously shown. In fact several derivation trees may result in the same derived tree. For example, in Figure 1.7, another set of LTAG etrees and the resulting derivation tree are given for the same derived tree (fragment), due to different ways of extracting elementary trees. This property gives us the flexibility of locating features that are useful for SRL from different styles of LTAG derivation trees. We applied this property through all of our work.

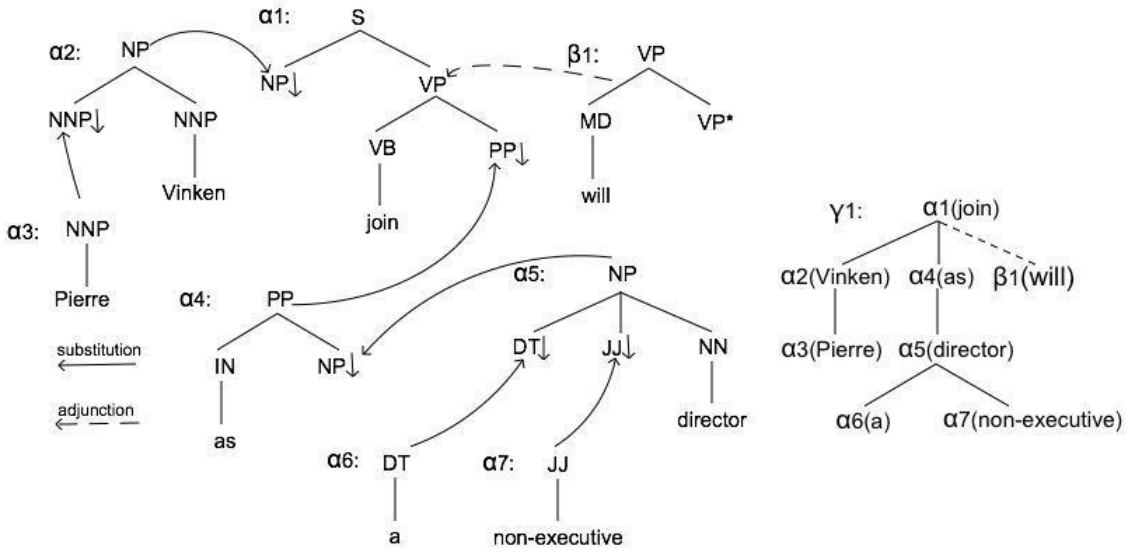

Figure 1.6: LTAG derivation tree $\gamma 1$ for the example.

1.4.4 Some Important Properties of LTAG

Each grammar formalism specifies a domain of locality, i.e., a domain over which various dependencies (syntactic and semantic) can be specified. It turns out that the various properties of a formalism (syntactic, semantic, computational, and even psycholinguistic) follow, to a large extent, from the initial specification of the domain of locality.

In a context-free grammar (CFG) the domain of locality is the one level tree corresponding to a grammar rule in a CFG. Compared to CFG, the elementary trees of an LTAG are minimal syntactic and semantic units since dependencies (such as predicate-argument) are supposed to be localized within the elementary trees. There are two key properties of LTAG: (i) Extended Domain of Locality (EDL), which allows (ii) Factoring Recursion from the domain of Dependencies (FRD), thus attempting to make all predicate-argument dependencies local (Joshi and Schabes, 1997).

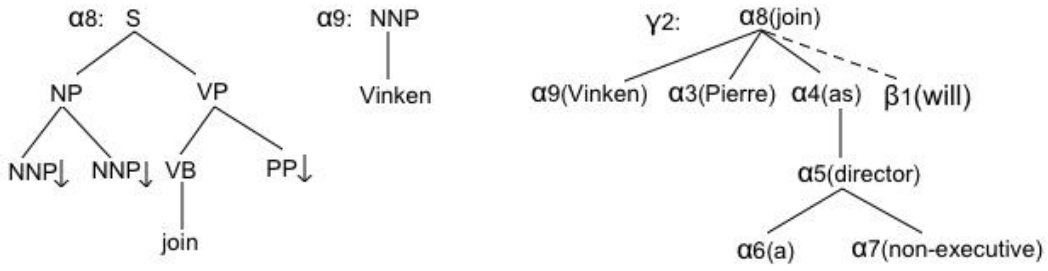

Figure 1.7: Compared to etrees in Figure 1.3, another plausible set of LTAG elementary trees (new etrees α_8, α_9) and the derivation tree γ_2 obtained by composing the elementary trees.

1.5 Lexicalized Tree-Adjoining Grammars and Semantic Role Labeling

SRL describes the dependencies between a particular predicate and its arguments in a sentence. These dependencies are expected to be represented within some domain of locality around the predicate in a parse tree. Current SRL features that are commonly used in the community are also defined aiming to capture such dependencies from the parse tree.

This notion coincides with the LTAG property of EDL. This property also allows factoring of selected portions of the parse in order to provide a local dependency between predicate and argument even in cases where they might be separated in a parse tree. In the SRL setting, it means we can simply focus on LTAG elementary trees and ignore various modal auxiliaries or negation modifiers in the parse tree, whose SRLs (if any) have little connection with the arguments of the predicate. Factoring away this type of recursion not only makes the dependencies more local, but can also facilitate SRL for modal verbs (AM-MOD) and negation (AM-NEG). For instance, in Figure 1.3, with LTAG we can factor away the auxiliary etree β_1 (recursion of *VP*s) to provide a natural domain of locality for the predicate *join* and its arguments without losing important information for SRL prediction. The role label AM-MOD for *MD(will)* does not depend on the specific predicate *join* and removing this auxiliary etree creates a local relation between the predicate and arguments like A0, A1, etc.

Based on the discussion above, we can see that LTAG and SRL are related closely. Ideally, the LTAG elementary tree that is anchored with a predicate should automatically form a frame which contains all the core arguments (mandatory arguments) of the predicate. The features that accompany the elementary tree are more salient for determining semantic role labels in that they are extracted from a deeper syntactic structure and therefore more linguistic motivated. In our work, we are interested in not only the LTAG elementary trees, but also the LTAG derivation trees, because the LTAG derivation trees contain information such as typological relations between predicate elementary trees and argument elementary trees, which is important to the SRL task. In Chapter 2, 3 and 4, we will present our work of using LTAG derivation trees for the SRL task.

Chapter 2

LTAG-Spinal for SRL

LTAG-spinal is a novel variant of traditional Lexicalized Tree Adjoining Grammar (LTAG) introduced by (Shen, 2006). The LTAG-spinal Treebank (Shen, Champollion, and Joshi, 2008) combines elementary trees extracted from the Penn Treebank with PropBank annotation. In this work, we present a semantic role labeling (SRL) system based on this new resource and provide an experimental comparison with CCGBank (Gildea and Hockenmaier, 2003) and a state-of-the-art SRL system based on Treebank phrase-structure trees. Deep linguistic information such as predicate-argument relationships that are either implicit or absent from the original Penn Treebank are made explicit and accessible in the LTAG-spinal Treebank, which we show to be a useful resource for semantic role labeling.

2.1 Introduction

The LTAG-spinal formalism was initially proposed for automatic treebank extraction and statistical parsing (Shen and Joshi, 2005). However, its PropBank-guided treebank extraction process further strengthens the connection between the LTAG-spinal and semantic role labeling. In particular, the development of the LTAG-spinal parsers make the LTAG-spinal derivation trees become immediately available for the SRL task.

Our experimental results have shown that our LTAG-spinal based SRL system achieves very high precision on both gold-standard and automatic parses, and significantly outperforms the one using CCGbank. More importantly, it shows that LTAG-spinal is an useful

resource for semantic role labeling, with the potential for further improvement.

2.2 LTAG-spinal, its Treebank, and Parsers

This section gives a brief introduction of the LTAG-spinal formalism, its Treebank that is extracted with the help of PropBank annotation, and its two statistical parsers that are trained on the Treebank. Predicate-argument relations encoded in the LTAG-spinal treebank will also be discussed to illustrate its compatibility with PropBank and their potential utility for the SRL task.

2.2.1 LTAG-spinal

The LTAG-spinal formalism (Shen, Champollion, and Joshi, 2008) is a variant of LTAG. Compared to traditional LTAG, the two types of elementary trees, initial and auxiliary trees, are in *spinal* form with no substitution nodes for arguments appearing in the predicate etree: a spinal initial tree is composed of a *lexical spine* from the root to the anchor, and nothing else; a spinal auxiliary tree is composed of a *lexical spine* and a *recursive spine* from the root to the foot node. For example, in Figure 2.1 (from (Shen, Champollion, and Joshi, 2008)), the lexical spine for the auxiliary tree is $B_1, .., B_i, .., B_n$, the recursive spine is $B_1, .., B_i, .., B_1^*$. Two operations *attachment* and *adjunction* are defined in LTAG-spinal where adjunction is the same as adjunction in the traditional LTAG; attachment stems from *sister adjunction* as defined in Tree Insertion Grammar (TIG) (Schabes and Shieber, 1994), which corresponds to the case where the root of an initial tree is taken as a child of another spinal etree. The two operations are applied to LTAG-spinal etree pairs resulting in an LTAG derivation tree which is similar to a dependency tree (see Figure 2.2). In Figure 2.2, etree anchored with *continue* is the only auxiliary tree; all other etrees are initial trees. The arrow is directed from parent to child, with the type of operation labeled on the arc. The operation types are: *att* denotes *attachment* operation; *adj* denotes *adjunction* operation. The sibling nodes may have different landing site along the parent spine. For example, among the child nodes of the *stabilize* etree, the *to* etree has VP as landing site; while *even* has S as landing site. Such information, on some level, turns out to be helpful in differentiating the semantic role played by the different child nodes.

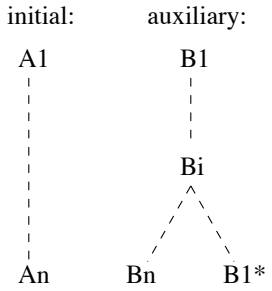

Figure 2.1: Spinal elementary trees

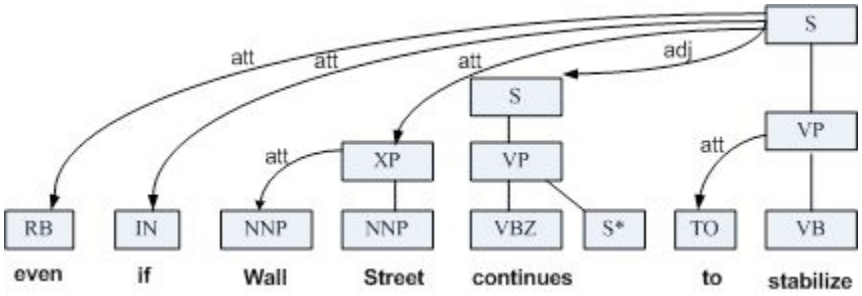

Figure 2.2: An example of LTAG-spinal sub-derivation tree, from LTAG-spinal Treebank Section 22

As claimed on page 14 of (Shen, 2006), "the domain of locality of LTAG is still maintained in LTAG-spinal in a way that syntactically dependent arguments are directly attached to the predicate. As a result, elementary trees are in the spinal form since arguments do not appear in the elementary tree of the predicate. This turns out to be a great advantage in handling coordination. In the traditional LTAG, one needs to transform the templates of predicate conjuncts in order to represent the shared arguments. However, representation of predicate coordination is rather easy with the spinal form". LTAG-spinal is weakly equivalent to traditional LTAG with adjunction constraints[1] (Shen, 2006).

So far, we can see that in contrast with traditional LTAG, where arguments refer to obligatory constituents only, subcategorization frames and argument-adjunct distinction

[1]null adjunction (NA), obligatory adjunction (OA) and selective adjunction (SA)

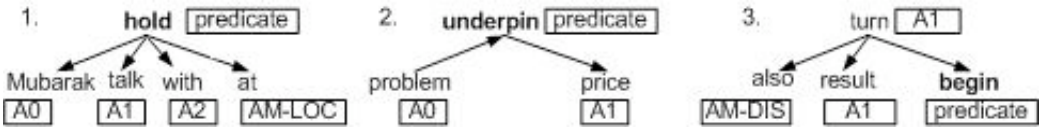

Figure 2.3: Three examples of LTAG-spinal derivation trees where predicates and their PropBank style argument labels are given. These examples are from LTAG-spinal Treebank Section 22.

are underspecified in LTAG-spinal. Since argument-adjunct disambiguation is one of the major challenges faced by LTAG treebank construction, LTAG-spinal works around this issue by leaving the disambiguation task for further deep processing, such as semantic role labeling. In this sense, LTAG-spinal actually shares the common ground with our previous work in extracting LTAG elementary trees.

The LTAG-spinal Treebank is extracted from the Penn Treebank by exploiting PropBank annotation. Specifically, as described in (Shen, Champollion, and Joshi, 2008), a Penn Treebank syntax tree is taken as an LTAG-spinal derived tree; then, information from the Penn Treebank and PropBank is merged using tree transformations. For instance, LTAG predicate coordination and instances of adjunction are recognized using PropBank annotation. LTAG elementary trees are then extracted from the transformed Penn Treebank trees recursively using the PropBank annotation and a Magerman-Collins style head percolation table.

This guided extraction process allows syntax and semantic role information to be combined in LTAG-spinal derivation trees. For example, the Penn Treebank does not differentiate raising verbs and control verbs; however, based on the PropBank information, LTAG-spinal makes this distinction explicit. Thus, the error of taking a subject argument which is not semantically an argument of the raising verb can be avoided. Another property of LTAG-spinal Treebank extraction lies in the flexibility and simplicity of the treatment of predicate coordination (see (Shen, Champollion, and Joshi, 2008)). Figure 2.3 shows three examples of PropBank annotation as decorations over the LTAG-spinal derivation trees. In each derivation tree, each node is associated with LTAG-spinal etrees. Each argument (A0, A1, *etc.*) is referred to as A and the predicate is called P. In most cases, the argument is

found locally in the derivation tree due to the extended domain of locality in etrees. Thus, most arguments are identified by the pattern $P \rightarrow A$ or $P \leftarrow A$. The next section contains a discussion of such patterns in more detail.

Two statistical parsers have been developed by Libin Shen specifically for training on the LTAG-spinal treebank: a left-to-right incremental parser (Shen and Joshi, 2005) and a bidirectional incremental parser (Shen and Joshi, 2008). If one compares the output of these two parsers, the left-to-right parser produces full LTAG-spinal derivation trees (including all the information about specific elementary trees used in the derivation and the attachment information within the etrees) while the bidirectional parser produces derivation trees without information about elementary trees or attachment points (similar to output from a dependency parser). In this work, we use the left-to-right incremental parser for its richer output because our SRL system uses feature functions that use information about the elementary trees in the derivation tree and the attachment points between etrees. The landing site of the child node along the parent spine is useful for identifying different types of arguments in SRL. For example, assume the parent spine is "S-VP-VB-anchor" (the root label is S, and "anchor" is where the lexical item is inserted). Along with direction information, the landing site label "S" is likely to be a good indicator for argument A0 (subject) while the landing site label "VP" could be a good indicator for "A1" (object). In this sense, the incremental left-to-right parser is preferable for semantic role labeling. However, having been developed earlier than the bidirectional parser, the incremental parser obtains 1.2% less in dependency accuracy compared to the bidirectional parser (Shen and Joshi, 2008).

2.2.2 Predicate-Argument Relations in the LTAG-spinal Treebank

The PropBank-guided extraction process for LTAG-spinal treebank naturally creates a close connection between these two resources. To examine the compatibility of the LTAG-spinal Treebank with PropBank, (Shen, Champollion, and Joshi, 2008) provides the frequency for specific types of paths from the predicate to the argument in the LTAG-spinal derivation trees from the LTAG-spinal Treebank. The 8 most frequent patterns account for 95.5% of the total predicate-argument pairs of the LTAG-spinal Treebank, of which 88.4% are directly connected pairs. These statistics not only provide empirical justification for

	Path Pattern	Number	Percent
1	P→A	8294	81.3
2	P←A, V←A	720	7.1
3	P←Px→A	437	4.3
4	P←Coord→Px→A	216	2.1
5	P←Ax←Py→A	84	0.82
6	P←Coord←Px→A	40	0.39
7	P←Px←Py→A	13	0.13
total recovered w/ patterns		9804	96.1
total		10206	100.0

Table 2.1: Distribution of the 7 most frequent predicate-argument pair patterns in LTAG-spinal Treebank Section 22. *P*: predicate, *A*: argument, *V*: modifying verb, *Coord*: predicate coordination.

the notion of the extended domain of locality (EDL) in LTAG-spinal (Shen, Champollion, and Joshi, 2008), they also provide motivation to explore this Treebank for the SRL task.

We collected similar statistics from Treebank Section 22 for the SRL task, shown in Table 2.1, where 7 instead of 8 patterns suffice in our setting. Each pattern describes one type of P(redicate)-A(rgument) pair with respect to their dependency relation and distance in the LTAG-spinal derivation tree. The reason that we combine the two patterns P←A and V←A into one is that from the SRL perspective, they are equivalent in terms of the dependency relation and distance with predicates. Each token present in the patterns – such as P, Px, Py, V, A, Ax and Coord – denotes a spinal etree in the LTAG-spinal derivation tree.

To explain the patterns more clearly, take the LTAG-spinal sub-derivation tree in Figure 2.2 as an example, and assume P(redicate) in question is *stabilize* then (*stabilize* → *even*), (*stabilize* → *if*), (*stabilize* → *Street*), (*stabilize* → *continue*), (*stabilize* → *to*) all belong to pattern 1; but only (*stabilize* → *Street*) is an actual predicate-argument pair. Similarly, when taking *continue* as P, the predicate-argument pair (*continue* ← *stabilize*) belongs to pattern 2, where *stabilize* corresponds to A(rgument) in the pattern; (*continue*, *Street*) in (*Street* ← *stabilize* → *continue*) is an example of pattern 3, where *stabilize* corresponds to Px and *Street* corresponds to A in the pattern 3 schema. Pattern 4 denotes the case where argument (A) is shared between coordinated predicates (P and Px). The main difference among patterns 5-7 is where the sibling node of A(rgument) is categorized into: predicate

(Px) in pattern 7, predicate coordination node (Coord) in pattern 6, and others (Ax) in pattern 5. We will retain this difference instead of merging it since the semantic relation between P and A varies based on these differences. Example sentences for other (rarer) patterns can be found in (Shen, Champollion, and Joshi, 2008).

2.3 LTAG-spinal based SRL System

In this section, we describe our LTAG-spinal based SRL system. So far, we have studied the LTAG-spinal formalism, its treebank, and parsers. In particular, the frequency distribution of the seven most-seen predicate-argument pair patterns in LTAG-spinal Treebank tells us that predicate-argument relationships typical to semantic role labeling are often local in LTAG-spinal derivation trees.

Pruning, argument identification and argument classification – the 3-stage architecture now standard in SRL systems is also used in this work. Specifically, for the sake of efficiency, nodes with high probability of being NULL (non-argument) should be filtered at the beginning; usually filtering is done based on some heuristic rules; after the pruning stage, argument identification takes place with the goal of classifying the pruning-survival nodes into argument and non-argument; for those nodes that have been classified as arguments, argument classification component will further label them with different argument types, such as A0, A1, *etc.*. Argument identification and classification are highly ambiguous tasks and are usually accomplished using a machine learning method.

For our LTAG-spinal based SRL system, we first collect the argument candidates for each predicate from the LTAG-spinal derivation tree. For each candidate, features are extracted to capture the predicate-argument relations. Binary classifiers for identification and classification are trained using SVMs and combined in a one-vs-all model. The results are evaluated using precision/recall/f-score.

2.3.1 Candidate Locations for Arguments

In SRL systems that perform role labeling of constituents in a phrase-structure tree, statistics show that after pruning, $\sim$98% of the SRL argument nodes are retained in the gold-standard trees in the Penn Treebank, which provides a high upper-bound for the recall of

the SRL system. Pruning away unnecessary nodes using a heuristic makes learning easier as well, as many of the false positives are pruned away leading to a more balanced binary classification problem during the semantic role identification and classification steps. We need a similar heuristic over LTAG-spinal nodes that will have high coverage with respect to SRL arguments and provide a high upper-bound for recall.

As previously shown, the seven most frequent predicate-argument pair patterns that are used to describe the specific types of paths from the predicate to the argument account for ∼96% of the total number of predicate-argument pairs in the LTAG-spinal Treebank. These patterns provide a natural candidate selection strategy for our SRL.

Table 2.2 shows a similar oracle test applied to the output of the LTAG-spinal parser on Section 22. The total drop in oracle predicate-argument identifiation drops 10.5% compared to gold-standard trees. 9.8% is lost from patterns 1 and 2. If we exclude those pairs that belong to pattern i in treebank but belong to pattern j ($i \neq j$) in automatic parses (so the pattern exists but is the wrong one for that constituent), the number drops to 81.6% from 85.6%. This indicates that in terms of the impact of the syntactic parser errors for SRL, the LTAG-spinal parser will suffer even more than the phrase structure parser. An alternative is to exhaustively search for predicate-argument pairs without considering patterns, which we found introduces too much noise in the learner to be feasible. Thus, the predicate-argument pairs selected through this phase are considered as argument candidates for our SRL system.

	Path Pattern	Number	Percent
1	P→A	7441	72.9
2	P←A, V←A	583	5.7
3	P←Px→A	384	3.8
4	P←Coord→Px→A	180	1.76
5	P←Ax←Py→A	75	0.73
6	P←Coord←Px→A	48	0.47
7	P←Px←Py→A	22	0.21
total recovered w/ patterns		8733	85.6
total		10206	100.0

Table 2.2: Distribution of the 7 patterns in LTAG-spinal parser output for Section 22.

2.3.2 Features

Based on the patterns, features are defined on predicate-argument pairs from LTAG derivation tree, mainly including *predicate etrees*, *argument etrees*, *intermediate etrees* and their "topological relationships" such as *operation, spine node, relative position* and *distance*. The following are the specific features used in our classifiers:

Features from predicate etree and its variants predicate lemma, POS tag of predicate, predicate voice, spine of the predicate etree, 2 variants of predicate etree: replacing anchor in the spine with predicate lemma, replacing anchor POS in the spine with voice. In Figure 2.2, if we take *stabilize* as predicate, these two variants are *S-VP-VB-stabilize* and *S-VP-VB-active* respectively.

Features from argument etree and its variants argument lemma, POS tag of argument, Named Entity (NE) label of the argument, spine of the argument etree, 2 variants of argument etree: replacing anchor in the spine with argument lemma, replacing anchor POS with NE label if any, label of root node of the argument spine. In Figure 2.2, if take *stabilize* as predicate, and *Street* as argument, the two variants are *XP-NNP-street* and *XP-ORGANIZATION*[2] respectively.

PP content word of argument etree if the root label of the argument etree is PP, anchor of the last daughter node. NE variant of this feature: replace its POS with the NE label if any.

Features from the spine node (SP1) The spine node is the landing site between predicate etree and argument etree. Features include the index along the host spine[3], label of the node, operation involved (*att* or *adj*).

Relative position of predicate and argument in the sentence: before/after.

Order of current child node among its siblings. In pattern 1, predicate etree is parent, and argument etree is child. This feature refers to the order of argument etree among its siblings nodes (with predicate etree as parent).

Distance of predicate etree and argument tree in the LTAG derivation tree: For example,

[2]XP-NNP is a normalized etree form used in (Shen, Champollion, and Joshi, 2008) for efficiency and to avoid the problem of sparse data over too many etrees.

[3]it can either be predicate etree or argument etree. For example, for pattern P←A, the A(rgument) etree is the host spine.

for pattern 1 and 2, the distance has value 0; for pattern 3, the distance has value 1.
Pattern ID valued 1-7. (see Table 2.1 and Table 2.2)
Combination of position and pattern ID, combination of distance and pattern ID, combination of position and order.
Features from intermediate predicate etree same features as predicate etree features.
Features from spine node of intermediate predicate etree and argument etree (SP2) for predicate-argument pairs of pattern 3-7. These features are similar to the SP1 features but instead between intermediate predicate etree and argument etree.
Relative position between predicate etree and intermediate etree.
Combination relative positions of argument etree and intermediate predicate etree + relative position of argument etree and predicate etree.

The features listed above are used to represent each candidate constituent (or node) in the LTAG-spinal derivation tree in training and test data. In both cases, we identify SRLs for nodes for each predicate. In training, each node comes with the appropriate semantic role label, or NULL if it does not have any (for the predicate). In the test data, we first identify nodes as arguments using these features (ARG v.s. NULL classification) and then classify a node identified as an argument with the particular SRL using one-vs-all binary classification.

2.4 Experiments

2.4.1 Data Set

Following the usual convention for parsing and SRL experiments, LTAG-spinal Treebank Section 2-21 is used for training and Section 23 for testing. The PropBank argument set is used which includes numbered arguments A0 to A5 and 13 adjunct-like arguments[4]. The reason for using an extended set of argument set in this work is to have a fair comparison with the previous work on CCGBank, which will be discussed later. 454 sentences in the Penn Treebank are skipped[5] from the LTAG-spinal Treebank (Shen, Champollion, and

[4]{AM-ADV, AM-CAU, AM-NEG, AM-DIR, AM-DIS, AM-LOC, AM-MNR, AM-MOD, AM-PNC, AM-PRD, AM-TMP, AM-REC, AM-EXT}

[5]Based on (Shen, Champollion, and Joshi, 2008), the 454 skipped sentences amount to less than 1% of the total sentences. 314 of these 454 sentences have gapping structures. Since PTB does not annotate

Joshi, 2008), which results in 115 predicate-argument pairs ignored in the test set.

We applied SVM-light (Joachims, 1999) with default linear kernel to feature vectors. 30% of the training samples are used to fine tune the regularization parameter c and the loss-function cost parameter j for both argument identification and classification. With parameter validation experiments, we set $c = 0.1$ and $j = 1$ for {A0, AM-NEG}, $c = 0.1$, $j = 2$ for {A1, A2, A4, AM-EXT} and $c = 0.1$ and $j = 4$ for the rest.

For comparison, we also built up a standard 3-stage phrase-structure based SRL system, where exactly the same data set[6] is used from 2004 February release of the PropBank. SVM-light with a linear kernel is used to train on a standard feature set as shown in Table 3.1. The Charniak and Johnson's parser (2005) is used to produce the automatic parses. Similar to before, both the missing constituents and discontinuous constituents participate in evaluation.

2.4.2 Results

We compared our LTAG-spinal based SRL system with a phrase-structure based one (see the description in earlier sections), for argument identification and classification. In order to analyze the impact of errors in syntactic parsers, results are presented on both gold-standard trees and automatic parses. Based on the fact that nearly 97% etrees that correspond to the core arguments[7] belong to pattern 1 and 2, which accounts for the largest portion of argument loss in automatic parses, the classification results are also given for these core arguments. We also compare with the CCG-based SRL presented in (Gildea and Hockenmaier, 2003)[8], which has a similar motivation as our work, except they use the Combinatory Categorial Grammar formalism and the CCGBank syntactic Treebank which was converted from the Penn Treebank.

---

the trace of deleted predicates, additional manual annotation is required to handle these sentences. For the rest of the 146 sentences, abnormal structures are generated due to tagging errors.

[6]The same 454 sentences are ignored.

[7]A0, A1, A2, A3, A4, A5

[8]Their data includes the 454 sentences. However, the missing 115 predicate-argument pairs account for less than 1% of the total number of predicate-argument pairs in the test data, so even if we award these cases to the CCGBank system the system performance gap still remains.

Scoring strategy To have a fair evaluation of arguments between the LTAG-spinal dependency parse and the Penn Treebank phrase structure, we report the *root/head-word* based scoring strategy for performance comparison, where a case is counted as positive as long as the root of the argument etree is correctly identified in LTAG-spinal and the head word of the argument constituent is correctly identified in phrase structure. In contrast, boundary-based scoring is more strict in that the string span of the argument must be correctly identified in identification and classification.

Results from using gold standard trees Table 2.3 shows the results when gold standard trees are used. We can see that with gold-standard derivations, LTAG-spinal obtains the highest precision on identification and classification; it also achieves a competitive f-score (highest f-score for identification) with the recall upper-bound lower by 2-3% than phrase-structure based SRL. However, the recall gap between the two SRL systems gets larger for classification compared to identification[9], which is due to the low recall that is observed with our LTAG-spinal based SRL based on our current set of features. If we compare the difference between the root/head-word based score and the boundary based score in the 3 scenarios, we notice that the difference reflects the discrepancy between the argument boundaries. It is not surprising to see that phrase-structure based SRL has the best match. However, CCGBank appears to have a large degree of mismatch. In this sense, root/head word based scoring provides fair comparison between LTAG-spinal SRL system and the CCGBank SRL system.

Recent work (Boxwell and White, 2008) changes some structures in the CCGBank to correspond more closely with the PropBank annotations. They also resolve split arguments that occur in PropBank and add these annotations into a revised version of the CCGBank. As a result they show that the *oracle* f-score improves by over 2 points over the (Gildea and Hockenmaier, 2003) oracle results for the numbered arguments only (A0, ..., A5). It remains an open question whether a full SRL system based on a CCG parser trained on this new version of the CCGBank will be competitive against the LTAG-spinal based and phrase-structure based SRL systems.

Results from using automatic parses Table 2.4 shows the results when automatic parses are used. With automatic parses, the advantage of LTAG-spinal in the precision

[9]No NULL examples are involved when training for argument classification.

Identification	gold-standard trees (p/r/f%)		
Scoring	LTAG	phrase	CCG
Root/head-word	**96.0**/92.1/**94.0**	93.0/94.0/93.5	n/a
classification (core)	gold-standard trees (p/r/f%)		
Scoring	LTAG	phrase	CCG
Root/head-word	**90.6**/83.4/86.9	87.2/88.4/87.8	82.4/78.6/80.4
classification (full)	gold-standard trees (p/r/f%)		
Scoring	LTAG	phrase	CCG
Root/head-word	**88.2**/81.7/84.8	86.1/87.1/86.6	76.3/67.8/71.8
Boundary	**87.4**/81.0/84.1	86.0/87.0/86.5	67.5/60.0/63.5

Table 2.3: Using gold standard trees: comparison of the three SRL systems for argument identification, core and full argument classification

scores still exists, giving a higher score in both identification and core argument classification: only 0.5% lower for full argument classification. However, with over 6% difference in upper-bound of recall ($\leq$85.6% from LTAG-spinal; $\sim$91.7% from Charniak's parser), the gap in recall becomes larger: increasing to $\sim$10% in automatic parses from $\sim$6% in gold-standard trees.

The identification result is not available for CCG-based SRL. In terms of argument classification, it is significantly outperformed by the LTAG-spinal based SRL. In particular, it can be seen that the LTAG-spinal parser performs much better on argument boundaries than CCG-based one.

One thing worth mentioning is that since neither the LTAG-spinal parser nor Charniak's parser provides trace (empty category) information in their output, no trace information is used for LTAG-spinal based SRL or the phrase-structure based SRL even though it is available in their gold-standard trees. No function tag information is used either.

2.5 Conclusion and Future Work

With a small feature set, the LTAG-spinal based SRL system described in this work provides the highest precision in almost all the scenarios, which indicates that the shallow semantic relations, e.g., the predicate-argument relations that are encoded in the LTAG-spinal Treebank are useful for SRL, especially when compared to the phrase structure Penn

Identification	automatic parses (p/r/f%)		
Scoring	LTAG	phrase	CCG
Root/head-word	**85.8**/80.0/82.8	85.8/87.7/86.7	n/a
classification (core)	automatic parses (p/r/f%)		
Scoring	LTAG	phrase	CCG
Root/head-word	**81.0**/71.5/76.0	80.1/82.8/81.4	76.1/73.5/74.8
classification (full)	automatic parses (p/r/f%)		
Scoring	LTAG	phrase	CCG
Root/head-word	78.0/70.0/73.7	78.5/80.3/79.4	71.0/63.1/66.8
Boundary	72.3/65.0/68.5	73.8/75.5/74.7	55.7/49.5/52.4

Table 2.4: Using automatic parses: comparison of the three SRL systems for argument identification, core and full argument classification

Treebank. (Shen, Champollion, and Joshi, 2008) achieves an f-score of 91.6% for non-trace SRL identification on the entire Treebank by employing a simple rule-based system, which also suggested this conclusion. In other words, there is a tighter connection between the syntax and semantic role labels in the LTAG-spinal representation.

However, in contrast to the high precision, the recall performance of LTAG-spinal based SRL needs a further improvement, especially for the argument classification task. From the SRL perspective, on one hand, this may be due to the pattern-based candidate selection, which bounds above the number of predicate-argument pairs that can be recovered for SRL; on the other hand, it suggests that the features for argument classification need to be examined more carefully, compared to the feature selection for argument identification, especially for A2 and A3 (as indicated by our error analysis on the results on the development set). A possible solution is to customize a different feature set for each argument type during classification, especially for contextual information.

Experiments show that when following the pipelined architecture, the performance of LTAG-based SRL is more severely degraded by the syntactic parser, compared to the SRL using phrase structure and CCG formalism. Even though the left-to-right statistical parser that was trained and evaluated on the LTAG-spinal Treebank achieves an f-score of 89.3% for dependencies on Section 23 of this treebank (Shen and Joshi, 2005), which is not far from the f-score of 91.0% obtained on sentences of length 100 or less in Section 23 of Penn Treebank by Charniak and Johnson's parser (2005), the SRL that used this output is worse

than expected. An oracle test shows that via the same 7 patterns, only 81.6% predicate-argument pairs can be recovered from the automatic parses, which is a big drop from 96.1% when we use the LTAG-spinal Treebank trees. Parser accuracy is high overall, but needs to be more accurate in recovering the dependencies between predicate and argument.

The SRL suffers the low recall not only when the automatic parses are used, but also when the gold trees are used. Based on this, we would expect that a thorough error analysis and feature calibrating can give us a better idea in terms of how to increase the recall in both cases.

As a final note, we believe that our effort on using LTAG-spinal for SRL is a valuable exploration of the LTAG-spinal formalism and its Treebank resource. We hope our work will provide useful information on how to better utilize this formalism and the Treebank resource for semantic role labeling.

Chapter 3

LTAG based Features for SRL

Chapter 2 shows that the LTAG-spinal based SRL system has much lower recall than the phrase-structure based one when an LTAG parser is used to produce the LTAG derivation trees. In Chapter 2 we identified the issue of low recall to be caused by issues with using the LTAG parser for the SRL task. To take advantage of the high-performance phrase-structure parsers as well as the potential utility of LTAG formalism to the SRL task, we extract LTAG derivation trees from phrase-structure parse trees rather than produce them from an LTAG parser. In this chapter, we will introduce our work of using LTAG based features, which are extracted from such LTAG derivation trees, for semantic role labeling. Experiments show that LTAG-based features can improve SRL accuracy at a statistically significant level in comparison with the best known set of features used in the state-of-the-art SRL systems. To show the usefulness of the features, we provide an experimental study comparing LTAG-based features with the "standard" set of features[1] and kernel methods used in state-of-the-art SRL systems.

3.1 Motivation

SRL feature extraction has relied on various syntactic representations of input sentences, such as syntactic chunks (Hacioglu et al., 2004) and full syntactic parses (Gildea and

[1]Features that are widely used in state of the art SRL systems and extracted from phrase-structure parse trees. See Table 3.1 in Section 3.3.

Jurafsky, 2002). In contrast with features from shallow parsing, previous work (Gildea and Palmer, 2002; Punyakanok, Roth, and Yih, 2005b) has shown the necessity of full syntactic parsing for SRL. In order to generalize the *path* feature (see Table 3.1 in Section 3.3) which is probably the most salient (while being the most data sparse) feature for SRL, previous work has extracted features from other syntactic representations, such as CCG derivations (Gildea and Hockenmaier, 2003) and dependency trees (Hacioglu, 2004) or integrated features from different parsers (Pradhan et al., 2005). To avoid explicit feature engineering on trees, Moschitti (2004) used convolution kernels on selective portions of syntactic trees.

Most SRL systems exploit syntactic trees as the main source of features. We would like to take this one step further and show that using LTAG derivation trees as an additional source of features can improve both argument identification and classification accuracy in SRL.

3.2 Using LTAG-based Features in SRL

In Section 1.4 of Chapter 1, we gave an introduction of LTAG formalism and described the relations between this formalism and semantic role labeling. Here we assume the readers have the background knowledge and we will not go into details of this aspect.

As aforementioned, LTAG is a formal grammar and not a linguistic theory, so elementary trees can be defined in many ways. A reasonable way to define SRL features is to provide a strictly local dependency (i.e. within a single etree) between predicate and argument. There have been many different proposals on how to maintain syntactic locality (Xia, 1999; Chen and Vijay-Shanker, 2000) and SRL locality (Chen and Rambow, 2003; Shen and Joshi, 2005) when extracting LTAG etrees from a Treebank. These proposed methods are exemplified by the derivation tree γ_1 in Figure 3.1. However, in most cases they can only provide a local dependency between predicate and argument for 87% of the argument constituents (Chen and Rambow, 2003), which is too low to provide high SRL accuracy. In LTAG-based statistical parsers, high accuracy is obtained by using the Magerman-Collins head-percolation rules in order to provide the etrees (Chiang, 2000). This method is exemplified by the derivation tree γ_2 in Figure 3.1. Comparing γ_1 with γ_2 in Figure 3.1 and assuming that *join* is the predicate and the *NP* is the potential argument,

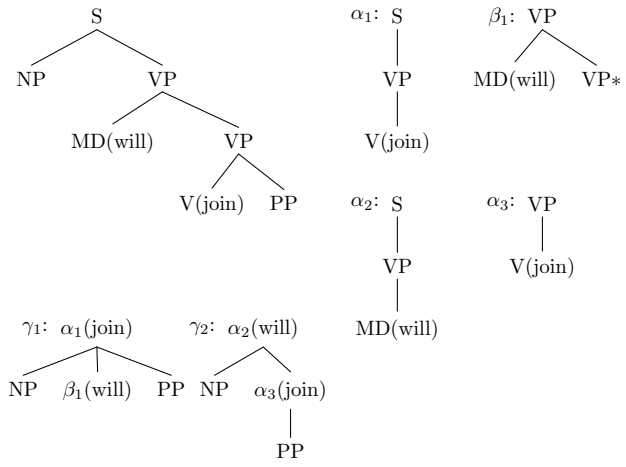

Figure 3.1: A parse tree schematic, and two plausible LTAG derivation trees for it: derivation tree γ_1 uses elementary trees α_1 and β_1 while γ_2 uses α_2 and α_3.

the path feature as defined over the LTAG derivation tree γ_2 is more useful for the SRL task as it distinguishes between main clause and non-finite embedded clause predicates. This alternative derivation tree also exploits the so-called *extended domain of locality* (Joshi and Schabes, 1997). In our method of obtaining LTAG derivation trees, we crucially rely on features defined on LTAG derivation trees of the latter kind. We use *sister adjunction* (See Section 2.2 for the definition) which is commonly used in LTAG statistical parsers to deal with the relatively flat Penn Treebank trees (Chiang, 2000).

We use polynomial kernels to create combinations of features defined on LTAG derivation trees.

3.2.1 LTAG-based Feature Extraction

In order to create training data for the LTAG-based features, we convert the Penn Treebank phrase structure trees into LTAG derivations. First, we prune the Treebank parse tree using certain constraints (Section 3.2.2 will give more details). Then we decompose the pruned parse trees into a set of LTAG elementary trees and obtain a derivation tree. For each constituent in question, we extract features from the LTAG derivation tree. We combine these features with the standard features used for SRL and train an SVM classifier on the combined LTAG derivation plus SRL annotations from the PropBank corpus.

For the test data, we report on results using the gold-standard Treebank data, and in addition we report results on automatically parsed data using Charniak's parser (Charniak, 2000) as provided by the CoNLL 2005 shared task. We did this for three reasons: (i) our results are directly comparable to those who have used Charniak's parses distributed with the CoNLL 2005 data-set; As one of the major benchmark data sets for the SRL task, CoNLL-2005 data is actually still commonly used within the SRL research community (Boxwell, Mehay, and Brew, 2009; Sun, Sui, and Wang, 2009); (ii) Although the new re-ranking model of Charniak and Johnson's parser (2005) (Charniak and Johnson, 2005) is reported to improve upon Charniak (2000) significantly, the overall difference in SRL accuracy of using these two models is only around 0.3-0.4% (Charniak and Johnson (2005) over Charniak (2000)), as reported in (Johansson and Nugues, 2008b); and (iii) the quality of LTAG derivation trees depends indirectly on the quality of head dependencies recovered by the parser and it is a well-known folklore result (see Table 3 in (McDonald, Crammer, and Pereira, 2005)) that applying the head-percolation heuristics on parser output produces better dependencies when compared to dependencies directly recovered by the parser (whether the parser is an LTAG parser or a lexicalized PCFG parser).

3.2.2 Pruning Parse Trees

Given a parse tree, the pruning component identifies the predicate in the tree and then only admits those nodes that are sisters to the path from the predicate to the root. It is commonly used in the SRL community (cf. (Xue and Palmer, 2004)) and our experiments show that 91% of the SRL targets can be recovered despite this aggressive pruning. We make two enhancements to the pruned Propbank tree: we enrich the sister nodes with head information, a part-of-speech tag and word pair: $\langle t, w \rangle$; and PP nodes are expanded to include the NP complement of the PP (including head information). Note that the target SRL node is still the PP. Figure 3.2 is a pruned parse tree for a sentence from the PropBank.

3.2.3 Decompositions of Parse Trees

After pruning, the pruned tree is decomposed around the predicate using standard head-percolation based heuristic rules[2] to convert a Treebank tree into an LTAG derivation tree. Figure 3.3 shows the resulting etrees after decomposition. Figure 3.4 is the derivation tree for the entire pruned tree. Each node in this derivation tree represents an etree in Figure 3.3. In our model we make an independence assumption that each SRL is assigned to each constituent independently, conditional only on the path from the *predicate etree* to the *argument etree* in the derivation tree. Different etree siblings in the LTAG derivation tree do not influence each other in our current models.

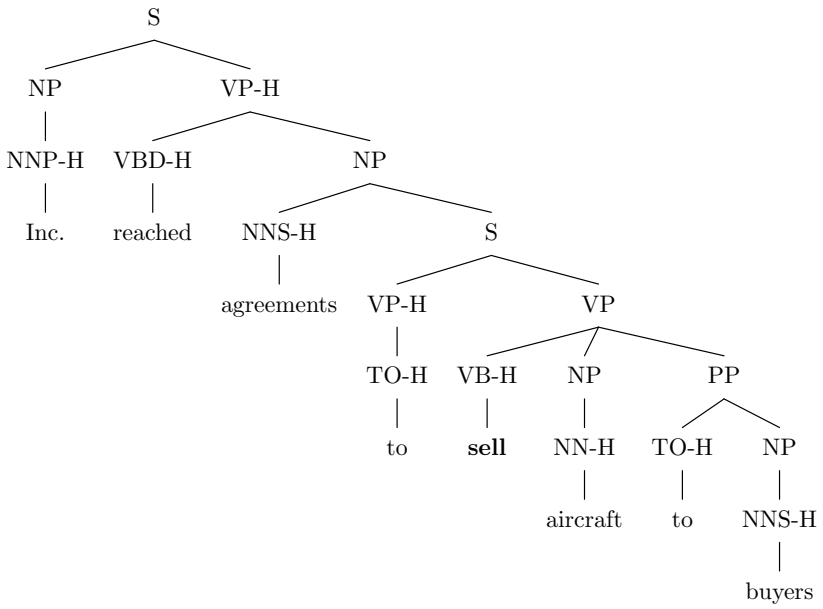

Figure 3.2: The pruned tree for the sentence "*Ports of Call Inc. reached agreements to* ***sell*** *its remaining seven aircraft to buyers that weren't disclosed.*"

[2] using `http://www.isi.edu/~chiang/software/treep/treep.html`

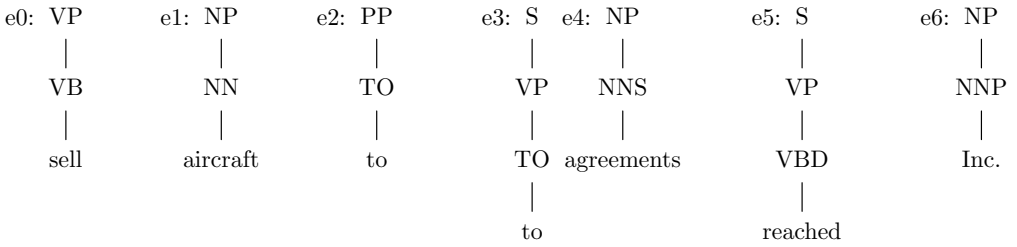

Figure 3.3: Elementary trees after decomposition of the pruned tree.

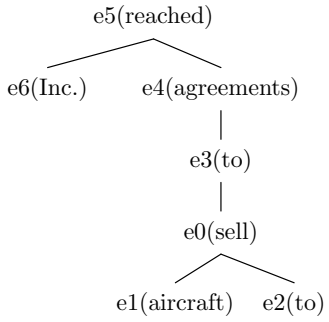

Figure 3.4: LTAG derivation tree for Figure 3.2.

3.2.4 LTAG-based Features

We defined 5 LTAG feature categories: *predicate etree*-related features (**P** for short), *argument etree*-related features (**A**), *subcategorization*-related features (**S**), *topological relation*-related features (**R**), *intermediate etree*-related features (**I**). Since we consider up to 6 intermediate etrees between the predicate and the argument etree, we use I-1 to I-6 to represent these 6 intermediate trees respectively.

Category P: Predicate etree & its variants *Predicate etree* is an etree with a predicate, such as $e0$ in Figure 3.3. This new feature complements the predicate feature in the standard SRL feature set. One variant is to remove the predicate lemma. Another variant is a combination of *predicate tree w/o predicate lemma&POS* and *voice*. In addition, this variant combined with predicate lemma comprises another new feature.

In the example, these three variants are *(VP(VB))* and *(VP)_active* and *(VP)_active_sell* respectively.

Category A: Argument etree & its variants Analogous to the predicate etree, the *argument etree* is an etree with the target constituent and its head. Similar to predicate etree related features, argument etree, argument etree with removal of head word, combination of *argument etree w/o head POS&head word* and *head Named Entity (NE) label (if any)* are considered. For example, in Figure 3.3, these 3 features for $e6$ are $e6$, *(NP(NNP))* and *(NP)_LOC* with head word "Inc." having NE label "LOC".

Category S: Index of current argument etree in subcat frame of predicate etree Sub-categorization is a standard feature that denotes the immediate expansion of the predicate's parent. For example, it is *V_NP_PP* for predicate *sell* in the given sentence. For argument etree $e1$ in Figure 3.3, the index feature value is 1 since it is the very first element in the (ordered) subcat sequence.

Category R:

Relation type between argument etree & predicate etree This feature is a combination of *position* and *modifying relation*. *Position* is a binary valued standard feature to describe if the argument is before or after the predicate in a parse tree. For each argument etree and intermediate etree, we consider three types of modifying relations they may have with the predicate etree: *modifying* (value 1), *modified* (value 2) and neither (value 3). From Figure 3.4, we can see $e1$ modifies $e0$ (predicate tree). So their modifying relation type value is 1; combining this value with the position value, this feature for $e1$ is "1_*after*".

Attachment point related features Label of the attachment point of argument etree and its parent and index of the attachment point along the parent etree. For $e1$ in the example, *VP* in the predicate tree is the label and 0 is the index of attachment point.

Distance This feature is the number of intermediate etrees between the argument etree and the predicate etree in the derivation tree. In Figure 3.4, the distance from $e4$ to the predicate etree is 1 since only one intermediate etree $e3$ is between them.

Category I:

Intermediate etree related features *Intermediate etrees* are those etrees that are located between the predicate etree and argument etrees. The set of features we propose

for each intermediate etree is quite similar to those for argument etrees except we do not consider the named-entity label for head words in this case.
Relation type of intermediate etree & predicate etree.
Attachment point of intermediate etree.
Distance between intermediate etree and predicate etree.

Up to 6 intermediate etrees are considered and the category I features are extracted for each of them (if they exist).

Each etree represents a linguistically meaningful fragment. The features of relation type and attachment point, as well as the distance characterize the topological relations among the relevant etrees. In particular, the attachment point and distance features can explicitly capture important information hidden in the standard path feature. The intermediate tree related features can give richer contextual information between predicate tree and argument trees. We added the **subcat index** feature to be complementary to the *sub-cat* and *syntactic frame* features in the standard feature set.

3.3 Standard Feature Set

Our standard feature set is a combination of features proposed by (Gildea and Jurafsky, 2002), (Surdeanu et al., 2003; Pradhan et al., 2004; Pradhan et al., 2005) and (Xue and Palmer, 2004). All features listed in Table 3.1 are used for argument classification in our baseline system; and features with asterisk are not used for argument identification[3]. We compare this baseline SRL with a system that includes a combination of these features with the LTAG-based features. Our baseline uses *all* features that have been used in the state-of-the-art SRL systems and as our experimental results show, these standard features do indeed obtain state-of-the-art accuracy on the SRL task. We will show that adding LTAG-based features can improve the accuracy over this very strong baseline.

[3]This is a standard idea in the SRL literature: removing features more useful for classification, e.g. named entity features, makes the classifier for identification more accurate.

Table 3.1: Standard features adopted by a typical SRL system. Features with asterisk $*$ are not used for argument identification.

Basic features from (Gildea and Jurafsky, 2002)
• **predicate lemma and voice** • **phrase type and head word** • **path from phrase to predicate** • **position:** phrase relative to predicate: before or after • **sub-cat** records the immediate structure that expands from predicate's parent
Additional features proposed by (Surdeanu et al. 2003; Pradhan et al., 2004, 2005)
• **predicate POS** • **head word POS** • **first/last word/POS** • **POS of word immediately before/after phrase** • **path length** • **LCA(Lowest Common Ancestor) path** from phrase to its lowest common ancestor with predicate • **punctuation immediately before/after phrase**$*$ • **path trigrams**$*$: up to 9 are considered • **head word named entity label** such as "PER, ORG, LOC"$*$ • **content word named entity label for PP parent node**$*$
Additional features proposed by (Xue and Palmer, 2004)
• **predicate_phrase type** • **predicate_head word** • **voice_position** • **syntactic frame**$*$

3.4 Experiments

3.4.1 Experimental Settings

Training data (PropBank Sections 2-21) and test data (PropBank Section 23) are taken from the CoNLL-2005 shared task[4][5]. All the necessary annotation information such as

[4] `http://www.lsi.upc.es/~srlconll/`

[5] The main difference of CoNLL-2005 shared task data and Propbank data is that the CoNLL data introduces two additional label types for arguments where R-ARGX is used to represent co-referential arguments and C-ARGX is used to represent a continuation phrase of multi-constituent arguments, while Propbank does not make such a distinction. Thus in CoNLL data, ARGX and R-ARGX are treated as different labels and evaluated separately. A script that converts Propbank annotations to CoNLL format

predicates, parse trees as well as Named Entity labels is part of the data. The argument set we consider is {A0, A1, A2, A3, A4, AM} where AM is a generalized annotation of all adjuncts such as AM-TMP, AM-LOC, etc., where PropBank function tags like TMP or LOC in AM-TMP, AM-LOC are ignored (a common setting for SRL, see (Xue and Palmer, 2004; Moschitti, 2004)). We chose these labels for our experiments because they have sufficient training/test data for the performance comparison and provide sufficient counts for accurate significance testing. However, we also provide the evaluation result on the test set for the full CoNLL-2005 task (all argument types).

We use SVM-light[6] (Joachims, 1999) with a polynomial kernel (degree=3) as our binary classifier for argument classification. We applied a linear kernel to argument identification because the training cost of this phase is extremely computationally expensive. We use 30% of the training samples to fine tune the regularization parameter c and the loss-function cost parameter j for both stages of argument identification and classification. With parameter validation experiments, we set $c = 0.258$ and $j = 1$ for the argument identification learner and $c = 0.1$ and $j = 4$ for the argument classification learner.

The classification performance is evaluated using *Precision/Recall/F-score* (p/r/f) measures. We extracted all the gold labels of A0-A4 and AM with the argument constituent index from the original test data as the "gold output". When we evaluate, we contrast the output of our system with the gold output and calculate the p/r/f for each argument type.

Our evaluation criteria are based on predicting the SRL for *constituents* in the parse tree which is similar to the evaluation used in (Toutanova, Haghighi, and Manning, 2005). However, we also evaluate those PropBank arguments which do not have a corresponding constituent in the gold parse tree or the automatic parse tree: the *missing constituent* case. We also evaluate *discontinuous* PropBank arguments using the notation used in the CoNLL-2005 data-set but we do not predict them. This contrasts with some previous studies where the problematic cases have been usually discarded or the largest constituents in the parse tree that almost capture the missing constituent cases are picked as being the correct answer. In addition to the constituent based evaluation, in Section 3.4.4 we also

is available as part of the shared task software (`http://www.lsi.upc.edu/~srlconll/soft.html`). In our system, we call C-ARGX discontinuous argument. We treat both R-ARGX and C-ARGX as separate argument labels.

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

	Gold Standard		Charniak's Parser	
	std	std+ltag	std	std+ltag
p(%)	95.66	96.79	87.71	89.11
r(%)	94.36	94.59	84.86	85.51
f(%)	95.00	**95.68**	86.26	**87.27**$*$

Table 3.2: Argument identification results on test data

provide the evaluation of our model on the CoNLL-2005 data-set.

Because the main focus of this work is to evaluate the impact of the LTAG-based features, we did not consider the frameset or a distribution over the entire argument set or apply any inference/constraints as a post-processing stage as most current SRL systems do. We focus our experiments on showing the value added by introducing LTAG-based features to the SRL task over and above what is currently used in SRL research.

3.4.2 Argument Identification

Table 3.2 shows results on argument identification (a binary classification of constituents into argument or non-argument). To fully evaluate the influence of the LTAG-based features, we report the identification results on both Gold Standard parses and on Charniak's parser output.

As we can see, after combining the LTAG-based features with the standard features, the f-score increased from 95.00% to 95.68% with Gold-standard parses; and from 86.26% to 87.27% with the Charniak's parses (a larger increase). We can see LTAG-based features help in argument identification for both cases. This result is better than (Xue and Palmer, 2004), and better on gold parses compared to (Toutanova, Haghighi, and Manning, 2005; Punyakanok, Roth, and Yih, 2005b).

3.4.3 Argument Classification

Based on the identification results, argument classification will assign the semantic roles to the argument candidates. For each argument of A0-A4 and AM, a "one-vs-all" SVM classifier is trained on both the standard feature set (std) and the augmented feature set (std+ltag). Table 3.3 shows the classification results on the Gold-standard parses with the gold argument identification; Table 3.4 and 3.5 show the classification results on

the Charniak's parser with the gold argument identification and the automatic argument identification respectively. Scores for multi-class SRL are calculated based on the total number of correctly predicted labels, the total number of gold labels, and the number of labels in our prediction for this argument set.

class	std(p/r/f)%		std+ltag(p/r/f)%	
A0	96.69	96.71	96.71	96.77
	96.70		**96.74**	
A1	93.82	93.30	97.30	94.87
	93.56		**96.07**	
A2	87.05	79.98	92.43	81.42
	83.37		**86.58**	
A3	94.44	68.79	97.69	73.41
	79.60		**83.33**	
A4	96.55	82.35	94.11	78.43
	88.89		85.56	
AM	98.41	96.61	98.67	97.88
	97.50		**98.27**	
multi-class	95.35	93.62	97.15	94.70
	94.48		**95.91**	

Table 3.3: Argument classification results on Gold-standard parses with gold argument boundaries.

3.4.4 Discussion

From the results shown in the tables, we can see that by adding the LTAG-based features, the overall performance of the systems is improved both for argument identification and argument classification.

Table 3.3 and 3.4 show that with gold argument identification, the classification for each class in {A0, A1, A2, A3, AM} consistently benefit from LTAG-based features. Especially for A3, LTAG-based features lead to more than 3% improvement. But for A4 arguments, the performance drops 3% in both cases. As we noticed in Table 3.5, which presents the argument classification results on Charniak's parser output with automatic argument identification, the prediction accuracy for classes A0, A1, A3, A4 and AM is improved, but drops a little for A2.

class	std(p/r/f)%		std+ltag(p/r/f)%	
A0	96.04	92.92	96.07	92.92
	94.46		**94.47**	
A1	90.64	85.71	94.64	86.67
	88.11		**90.48**	
A2	84.46	75.72	89.26	75.22
	79.85		**81.64**	
A3	87.50	62.02	87.10	68.35
	72.59		**76.60**	
A4	90.00	79.12	90.54	73.62
	84.21		81.21	
AM	95.14	85.54	96.60	86.51
	90.09		**91.27**	
multi-class	93.25	86.45	94.71	87.15
	89.72		**90.77**	

Table 3.4: Argument classification results on Charniak's parser output with gold argument boundaries

In addition, we also evaluated our feature set on the full CoNLL 2005 shared task. The overall performance using LTAG features increased from 74.41% to 75.31% in terms of f-score on the full argument set. Our accuracy is most closely comparable to the 77.03% accuracy achieved on the full task by (Toutanova, Haghighi, and Manning, 2008) where the features are explored on a single Penn Treebank parse tree and local classifiers are applied for the SRL task. The 79.44% accuracy obtained by the top system in CoNLL 2005 (Punyakanok, Roth, and Yih, 2005a) is not directly comparable since their system used the more accurate n-best parser output of (Charniak and Johnson, 2005). In addition their system also used global inference. Our focus in this work was to propose *new LTAG features* and to evaluate impact of these features on the SRL task.

We also compared our proposed feature set against predicate/argument features (PAF) proposed by (Moschitti, 2004). We conducted an experiment using the *SVM-light-TK-1.2 toolkit*[7]. The PAF tree kernel is combined with the standard feature vectors by a linear operator. With settings of Table 3.5, its multi-class performance (p/r/f)% is

[7] `http://ai-nlp.info.uniroma2.it/moschitti/TK1.2-software/Tree-Kernel.htm`

class	std(p/r/f)%		std+ltag(p/r/f)%	
A0	86.50	86.18	88.17	87.70
	86.34		**87.93**∗	
A1	78.73	83.82	88.78	85.22
	81.19		**86.97**∗	
A2	85.40	73.93	83.11	75.42
	79.25		79.08	
A3	85.71	60.76	85.71	68.35
	71.11		**76.06**∗	
A4	84.52	78.02	89.47	74.72
	81.15		**81.43**	
AM	80.47	82.11	83.87	81.54
	81.29		**82.69**∗	
multi-class	81.79	82.90	86.04	84.47
	82.34		**85.25**∗	

Table 3.5: Argument classification results on Charniak's parser output with automatic argument boundaries

83.09/80.18/81.61 with linear kernel and 85.36/81.79/83.53 with polynomial kernel (degree=3) over *std* feature vectors.

3.4.5 Significance Testing

To assess the statistical significance of the improvements in accuracy we did a two-tailed significance test on the results of both Table 3.2 and 3.5 where Charniak's parser outputs were used. We chose *SIGF*[8], which is an implementation of a computer-intensive, stratified approximate-randomization test (Yeh, 2000). The statistical difference is assessed on SRL identification, classification for each class (A0-A4, AM), and the full SRL task (overall performance). In Table 3.2 and 3.5, we labeled numbers under *std+ltag* that are statistically significantly better from those under *std* with asterisk. The significance tests show that for the identification and full SRL task, the improvements are statistically significant with p value of 0.013 and 0.0001 respectively at a confidence level of 95%. The significance test on each class shows that the improvement by adding LTAG-based features is statistically significant for class A0, A1, A3 and AM. Even though in Table 3.5 the performance of

[8] `http://www.coli.uni-saarland.de/~pado/sigf/index.html`

A2 appears to be worse it is not significantly so, and A4 is not significantly better. In comparison, the performance of PAF did not show significantly better than *std* with p value of 0.593 at the same confidence level of 95%.

3.4.6 Analysis of the LTAG-based Features

	full	**full−P**	**full−R**	**full−S**	**full−A**	**full−I**	**std**
id	90.5	90.6	90.0	90.5	90.5	90.1	89.6
A0	84.5	84.3	84.6	84.5	84.3	83.5	84.2
A1	89.8	90.1	89.4	89.3	89.6	89.3	88.9
A2	84.2	84.2	84.0	83.7	83.6	83.6	84.9
A3	76.7	80.7	75.1	76.0	75.6	76.7	78.6
A4	80.0	83.3	80.0	79.6	80.0	80.0	79.2
AM	82.8	83.3	82.9	82.8	82.6	83.1	82.4

Table 3.6: Impact of each LTAG feature category (**P, R, S, A, I** defined in Section 3.2.4) on argument classification and identification on CoNLL-2005 development set (WSJ Section 24). **full** denotes the full feature set, and we use $-\alpha$ to denote removal of a feature category of type α. For example, **full−P** is the feature set obtained by removing all **P** category features. **std** denotes the standard feature set.

We analyzed the drop in performance when a particular type of LTAG feature category is removed from the full set of LTAG features (we use the broad categories **P, R, S, A, I** as defined in Section 3.2.4). Table 3.6 shows how much performance is lost (or gained) when a particular type of LTAG feature is dropped from the full set.

These experiments were done on the development set from CoNLL-2005 shared task, using the provided Charniak's parses. All the SVM models were trained using a polynomial kernel with degree 3. It is clear that the **S, A, I** category features help in most cases and **P** category features hurt in most cases, including argument identification. It is also worth noting that the **R** and **I** category features help most for identification. This vindicates the use of LTAG derivations as a way to generalize long paths in the parse tree between the predicate and argument. Although it seems LTAG features have negative impact on prediction of A3 arguments on this development set, dropping the **P** category features can actually improve performance over the standard feature set. In contrast, for the prediction

of A2 arguments, none of the LTAG feature categories seem to help.

Note that since we use a polynomial kernel in the full set, we cannot rule out the possibility that a feature that improves performance when dropped may still be helpful when combined in a non-linear kernel with features from other categories. However, this analysis on the development set does indicate that overall performance may be improved by dropping the **P** feature category.

We adopt this calibration result on the test set, where we first attempted to drop the **P** feature category for the argument identification, and drop the LTAG feature category that has shown the most hurting. Since the **P** feature category seems to hurt the performance of most arguments on the development set, we also tried to drop it in each case of the test set. It turns out that, in comparison to the development set, the test set responds differently to each of these feature categories. Table 3.7 shows the result.

	full	**full−P**	**full−R**	**full−S**	**full−A**	**full−I**	**std**
id	87.27	87.24	–	–	–	–	86.26
A0	87.93	**88.05**	**88.00**	–	–	–	86.34
A1	86.97	86.88	–	–	–	–	81.19
A2	79.08	**80.32**	–	–	–	–	79.25
A3	76.06	75.79	–	–	–	–	71.11
A4	81.43	80.92	–	–	–	–	81.15
AM	82.69	80.75	80.78	–	–	80.34	81.29

Table 3.7: Result of argument identification and classification on CoNLL-2005 test set (WSJ Section 23) by adopting LTAG feature calibration result.

Table 3.7 shows that dropping the **P** category helps to boost the performance of both A0 and A2 on the test set; however, the results of feature calibration on the development set show that dropping **P** does not help on A2 and will hurt on A0; except in these two cases, dropping **P** has shown to be helpful overall for all other cases in classification as well as in identification. This appears very different on the test set. In addition, dropping **R** and **I** feature categories has shown to be helpful for AM on the development set but shown hurting on the test set. The **R** category appears to have a similar impact on A0 for both the development set and the test set. By dropping the **P** features on A0 and A2, the SRL

performance on the test set increases from 85.25% to 85.39%.

Due to the inconsistency of the feature analysis result between the development set and the test set, it is hard to tell specifically which individual feature category hurts most. What we can tell is that the SRL overall performance can be boosted on a statistically significant level by incorporating all of these LTAG-based features.

3.4.7 Analysis based on Support Vectors

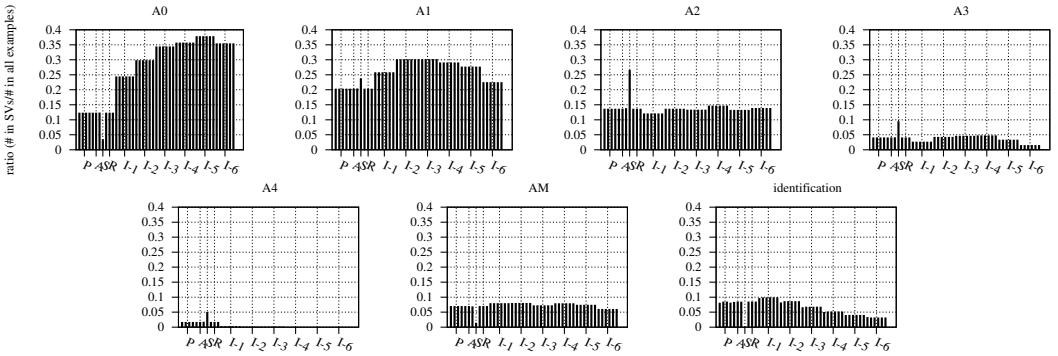

Figure 3.5: Illustration of impact of each LTAG feature category (**P, A, S, R, I-1, I-2, I-3, I-4, I-5, I-6** defined in Section 3.2.4) on argument classification and argument identification.

We also proposed another way to compare the effect that different LTAG feature categories have on this task. For each LTAG feature category, we count the occurrences of its instances in the Support Vectors (which can be obtained from the SVM classifier) and in all of the training data. Presumably the ratio between these two numbers indicates which feature category is predominantly useful in the large margin classification using SVMs. Figure 3.5 shows these ratios for the 10 feature categories on both argument classification (for each class) and argument identification. The x-axis denotes each feature category and y-axis denotes the ratio. Features in the **I** category play a more important role than those of **P** category and **A** category. For identification, intermediate etrees close to a predicate etree seem more important than those far from a predicate etree; for A0, it is the opposite; and for A1 and A3, those intermediate trees in the middle are more important. For A2 and AM, all **I** category features appear to contribute equally. The importance of the long

distance features using intermediate trees in the **I** category is interesting since it directly deals with the sparse data issue for standard parse path features. For A4, the **I** category has little impact and this class requires sparse lexical information for accurate classification. This also explains that why LTAG-based features did not help with A4 prediction in Table 3.3 and 3.4. The **S** feature category appears dramatically different between the group of A2, A3 and A4 and the group of A0, AM and identification. The importance of other feature categories such as **P, A** and **R** basically keeps on the same level in each case. The theoretical foundation of this method will be further studied in our future work.

3.5 Related Work

There has been some previous work in SRL that uses LTAG-based decomposition of the parse tree. Chen and Rambow (2003) use LTAG-based decomposition of parse trees (as is typically done for statistical LTAG parsing) for SRL. Instead of extracting a typical "standard" path feature from the derived tree, Chen and Rambow (2003) use the path within the elementary tree from the predicate to the constituent argument. Under this frame, they only recover semantic roles for those constituents that are localized within a single etree for the predicate, ignoring cases that occur outside the etree. As stated in their paper, "as a consequence, adjunct semantic roles (ARGM's) are basically absent from our test corpus"; and around 13% complement semantic roles cannot be found in etrees in the gold parses. In contrast, we recover all SRLs by exploiting more general paths in the LTAG derivation tree. A similar drawback can be found in (Gildea and Hockenmaier, 2003) where a **parse tree path** was defined in terms of Combinatory Categorial Grammar (CCG) types using grammatical relations between predicate and arguments. The two relations they defined can only capture 77% arguments in Propbank and they had to use a standard path feature as a replacement when the defined relations cannot be found in CCG derivation trees. In our framework, we use intermediate sub-structures from LTAG derivations to capture these relations instead of bypassing this issue.

Our approach shares similar motivations with the approach in (Shen and Joshi, 2005) which uses PropBank information to recover an LTAG treebank as if it were hidden data underlying the Penn Treebank. However their goal was to extract an LTAG grammar using PropBank information from the Treebank, and *not* the SRL task.

Features extracted from LTAG derivations are different and provide distinct information when compared to predicate-argument features (PAF) or sub-categorization features (SCF) used in (Moschitti, 2004) or even the later use of argument spanning trees (AST) in the same framework. The *adjunction* operation of LTAG and the extended domain of locality are not captured by those features as we have explained in detail in Section 1.4.

3.6 Conclusion

in this chapter we proposed to use LTAG derivation trees, which are extracted from the phrase-structure parse trees, as a novel source for SRL feature extraction and defined a new set of LTAG based features that have been shown to be useful for SRL task. In this work we show that LTAG-based features improve on the best known set of features used in current SRL prediction systems: the f-score for argument identification increased from 86.26% to 87.27% and from 82.34% to 85.25% for the SRL task, and the improvement is statistically significant. The analysis of the impact of each LTAG feature category shows that the intermediate etrees are important for the improvement.

Chapter 4

LTAG features for SRL using Latent SVMs

As shown previously, a phrase structure parse tree for a sentence can be generated by many different LTAG derivation trees. In this chapter, we use multiple LTAG derivation trees as latent features for semantic role labeling (SRL). We hypothesize that positive and negative examples of individual semantic roles can be reliably distinguished by possibly different latent LTAG features. For this reason we introduce latent support vector machines (LSVM) for the SRL task using these latent LTAG features. Our experiments on the PropBank-CoNLL'2005 dataset show that our method significantly outperforms the state-of -the-art SRL systems (even compared to models using global constraints or global inference over multiple parses). LSVMs are well suited for the SRL task. Our SRL system is composed of several individual component one-vs-all binary classifiers trained using our LSVM approach. Each individual binary classifier obtains near perfect precision.

4.1 Introduction

In Chapter 3 we have shown that the features that are extracted from the LTAG derivation trees, when combined with the standard feature set, can improve the SRL system performance on a statistically significant level. In that work, for a particular sentence, a single LTAG derivation tree was extracted from the Penn Treebank (PTB) parse tree based on

Magerman-Collins head percolation rules and *sister-adjunction* operation; and then used as a source of LTAG features for the argument prediction of all SRL instances of that sentence.

As mentioned in Section 1.4.2 of Chapter 1, an LTAG derivation tree is a composition of LTAG elementary trees via various operations and can be used to record the history of how a derived tree is generated. More than one linguistically plausible LTAG derivation can be extracted from a single PTB tree, as shown in Figure 1.6 and Figure 1.7. As additional examples, Figure 4.1 and Figure 4.2 also show two of the many possible sets of elementary trees for the derived tree in Figure 4.3. In this Chapter, we consider these multiple LTAG derivations as latent features for the SRL task. Since there are multiple LTAG derivations available for the single PTB tree, any single derivation tree extracted using one heuristic is not necessarily the best feature source for the SRL instance in question. By treating LTAG derivations as latent structures of a PTB tree, our method aims to find the most suitable LTAG derivation for each SRL instance. The work in this chapter is in fact an extension and generalization of the work in Chapter 3.

For each semantic role we hypothesize that positive and negative examples of individual semantic roles can be reliably distinguished by possibly different latent LTAG features, and for this reason we introduce a methodology that uses latent SVMs (LSVM) for SRL using latent LTAG features from multiple LTAG derivation trees. Our experiments show that LSVM seems to converge to a very predictive set of latent LTAG features for discriminative classification in most cases of SRL prediction. Our SRL system obtains a significant improvement in f-score compared to the use of a single LTAG derivation tree (using standard Magerman-Collins head rules). Our method gives the best reported precision when evaluated on the CoNLL-2005 shared task dataset, based on the PropBank corpus. Our method significantly outperforms the current top performing system on the CoNLL-2005 dataset in prediction of core arguments such as A0, A1, A2, and A3.

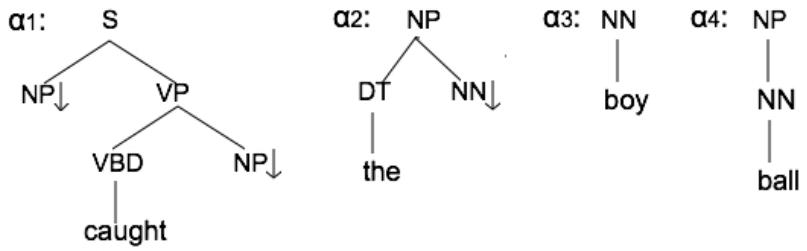

Figure 4.1: LTAG elementary trees resulting from one decomposition from the derived tree in Figure 4.3.

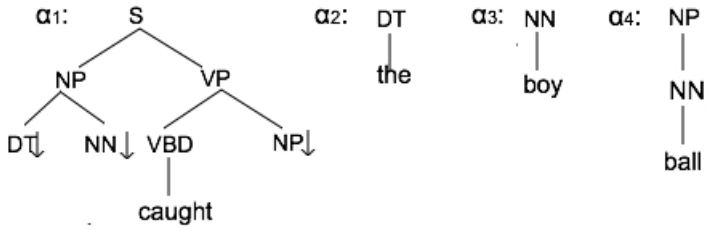

Figure 4.2: An alternative set of LTAG elementary trees resulting from another decomposition from the derived tree in Figure 4.3.

4.2 Generation of Latent LTAG Derivation Trees

In this section, we will first provide a review of LTAG derivation tree extraction from previous work, which is mainly used in the context of statistical parsing. Then we will describe the derivation trees we propose to generate for the SRL task.

In the context of statistical LTAG parsing, extracting LTAG derivation trees from the PTB trees has been extensively explored in (Chiang, 2000; Xia, 2001; Chen, 2002; Chiang and Bikel, 2002; Shen, 2004), which was mainly motivated by the following things.

1. Different from PCFG parsing, the sample space for LTAG parsing is the space of LTAG derivation trees, instead of the derived trees that are found in the PTB.

2. As we know, very large corpora are crucial to statistical natural language processing.

However no large corpora are immediately available for statistical LTAG parsing.

3. Given a CFG and head percolation scheme, an equivalent LTAG can be constructed whose derivations mirror the dependency analysis implicit in the head percolation scheme. Based on these facts, inducing an LTAG treebank (of LTAG derivation trees) from PTB has become one of the most important pre-processing steps for efficient statistical LTAG parsing.

To induce LTAG derivation trees from PTB derived trees, two types of ambiguities need to be resolved: *head competition* and *argument-adjuction distinction* (Shen, 2004).

4.2.1 Head Competition

Head-finding is the most commonly used method for grammar lexicalization, and has been proven to be effective in parsing. The so-called *Magerman-Collins* head percolation table is simply a deterministic, heuristic method to determine the head of each phrase given local syntactic context. The syntactic context that is used for a particular sentence is defined by the PTB bracketing for that sentence. For example, in figure 4.3, based on the heuristics VP is the head of rule S $\rightarrow$ NP VP; and VBD is the head of rule VP $\rightarrow$ VBD NP. Therefore the lexical head of VP(*caught*-VBD) is propagated to the parent nodes up to the root[1]. Following the lexical heads of a syntactic parse tree, we can get the spines of LTAG elementary trees. For example, in figure 4.3, the path from the lexical item *caught* to the root S forms the spine of the elementary tree anchored at *caught*; similarly, the path from the lexical item *boy* to NP(boy-NN) forms the spine of the elementary tree anchored at *boy*. If we specify the elementary trees in spinal form, then the head-finding rules essentially tell us how to generate LTAG elementary trees.

4.2.2 Argument-Adjunct Distinction

So far, the spine of each elementary trees can be generated using the head-percolation rules. For each node η in the parse tree, head-finding rules specify exactly one child of η as the head; then the remaining child nodes have to be disambiguated either as arguments

[1]Figure 4.3 (1) and (2) are equivalent in representing the heads. We used notation (1) here for ease of description. We will use notation (2) in the subsequent description.

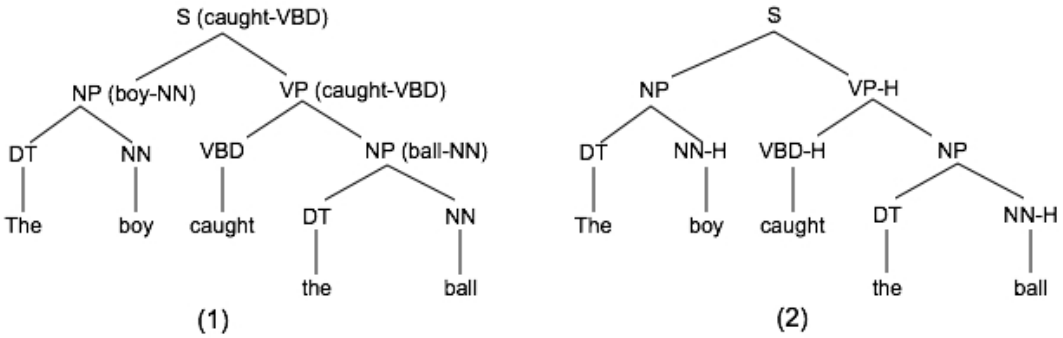

Figure 4.3: A simple lexicalized parse tree

or as adjuncts, which indicates the dependencies that need or need not be localized in the elementary trees. In (Xia, 2001; Chen, 2002), argument-adjunct distinction is solved with respect to the constituent label and function tags in the PTB trees as well as a set of pre-defined templates. Once this ambiguity is resolved, the operations (*substitution* or *adjunction* in standard TAG) for composing the elementary trees into LTAG derivation tree can also be determined accordingly.

4.2.3 Previous Work

Based on the ambiguities stated above, obviously many different sets of LTAG derivation trees could be extracted from the same PTB tree, by simply applying alternative sets of head-percolation rules. In practice some toolkits have been produced to reduce the amount of human effort needed for this process. *Treep*[2] (Chiang and Bikel, 2002) is such a tree processor that takes as input a sequence of phrase-structure trees and modifies their labels according to a set of head-percolation rules. Its envisioned purpose is as a front-end to the trainer of a statistical parser.

However, in previous work, almost identical head-lexicalizations were used either based on the *Magerman-Collins* head percolation table, or a slight variant of this table (Chen, 2002). The situation is similar for argument-adjunct disambiguation. In (Chiang, 2000), *sister adjunction* is used to deal with the relatively flat PTB trees for efficient statistical

[2] `http://www.isi.edu/~chiang/software/treep/treep.html`

parsing. In *sister-adjuction*, the root of the "sub etree[3]" can be added as a new daughter to any other node of the "main etree[4]".

The reason that all the work favors the lesser variety of LTAG elementary trees is that as stated in (Xia, 2001) "such transformation conforms to the general linguistic principles that guide the formation of handwritten TAGs" and "at least some of the dependencies are crucial in parsing disambuiguation". In other words, the lack of variety in LTAG derivation tree extraction in literature work has a strong motivation from parsing. In fact, (Shen, 2004) attempted to relax the linguistic constraints and extract the LTAG derivation trees nondeterministically using the expectation-maximization (EM) algorithm. However, the attempt was unsuccessful in terms of its utility for parsing.

4.2.4 Our Initiative

Previous work in LTAG derivation tree extraction was mainly motivated by parsing. If put it in a broader context by considering the SRL task, it would be interesting to explore the utility of other ways of LTAG derivation tree extraction. Based on our previous work, spinal-formed LTAG elementary trees are of more interest for SRL in terms of their simplicity and utility to feature representation. So is the *sister-adjunction* operation, which encodes the ambiguity of arguments and adjuncts that is a core aspect of the SRL task, and leaves the disambiguation to SRL. Besides, our argument candidate collecting does not depend on the local structure of elementary trees as in (Chen and Rambow, 2003). Instead, we follow the same approach as we did in Chapter 3. For SRL, we expect that the cost of generating larger LTAG elementary trees in a standard way will exceed the gain we get from their usefulness.

As far as SRL is concerned, each predicate is one of the "most important words" in a sentence, which means that there exists a phrase in the parse tree such that the predicate indicates the *aboutness* of the phrase. The most important words in a sentence, based on (Allen, 1994), are heads. Inspired by the fact that the predicate spine, when combined with other features, has turned out to be a useful feature for SRL prediction in our system, we would like to generate LTAG derivation trees using the head percolation rules that will

[3]The etree that is sister-adjoined to the other etree.

[4]The etree to which the sub etree is sister-adjoined.

always give higher priority to predicates for head determination, which does not always hold in *Magerman-Collins* head percolation table. Our experiments in this chapter will show that above strategy is successful in our setting for the SRL task. In the following, we will give more detail about how the derivation trees are generated based on the new head percolation rules.

4.2.5 LTAG Derivation Tree Extraction for SRL

In our LTAG derivation tree extraction, we exploit only *head competition* to create ambiguous LTAG derivations. For each node with a potential SRL we choose a daughter of that node as a potential head. In the resulting derivations, all the etrees are in spinal form and all sub etrees are attached to the main etree through *sister adjunction*.

For each *predicate*, we apply 3 heuristic head-percolation rules to generate 3 head-annotated parse trees per predicate:

1. Magerman-Collins rules.
2. Same as Rule 1, but label all consecutive VP nodes as head from the predicate until you see a non-VP node.
3. Same as Rule 1, but label all nodes as head from the predicate to the root of the PTB tree.

For each head-annotated parse tree, different predicate etrees are extracted using the algorithm given in (Chiang, 2000) that converts head-annotated parse trees into LTAG derivations. For each predicate etree, we produce argument etrees which localize the node that may receive a semantic role. Different argument etrees are created by exploiting head competition ambiguity for the SRL node in the etree, plus different ways to project to a root node that connects to the predicate etree. This process can produce multiple identical etrees, of which only one copy is retained. When the number of intermediate etrees (which are defined as etrees between predicate etree and argument etree in the LTAG derivation) is less than 3, we also consider all the possible variants for intermediate etrees based on the same strategy for generating argument etrees. One LTAG derivation is different from another if the combination of predicate etree, argument etree, and intermediate etrees (if

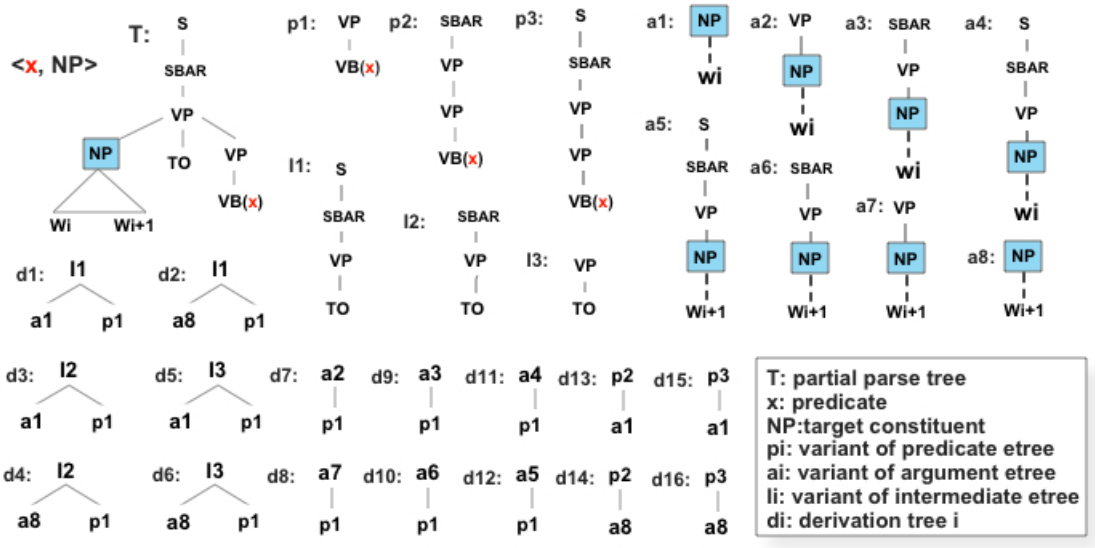

Figure 4.4: The latent derivation trees (d_i) extracted for predicate x and argument candidate NP in a parse tree fragment T. The extraction process works in the same fashion for more deeply embedded predicates (resulting in more nodes on the path from predicate to root node) and in cases where the argument candidate dominates any number of lexical items w_i. In contrast to (Chen and Rambow, 2003) argument nodes do not have to be the root node of an etree.

any) is different. Algorithm 1 shows the pseudo-code that is used to generate the LTAG derivation trees.

In our dataset on average we have ∼13000 derivation trees generated for each SRL instance. Figure 4.4 shows a worked out example to illustrate the above latent derivation tree generation process.

4.3 Latent Support Vector Machines (LSVMs)

Once we have a space of latent features and we are interested in learning an accurate set of classifiers for the SRL task we can sum over the different latent derivations in order to make a prediction using a log-linear model. This is potentially attractive but in this work we were drawn to a different methodology for the use of latent features. We hypothesize that for each SRL classification task, positive and negative examples of individual semantic

Algorithm 1 LTAG derivation trees generation

```
1: for each predicate in the sentence do
2:    generate the predicate elementary trees based on the 3 heuristic head-percolation
      rules as listed previously, and remove duplicates if any
3: end for
4: for each generated predicate elementary tree do
5:    for each target constituent (argument candidate) do
6:       for each lexical item that is dominated by the target constituent do
7:          take the lexical item as the anchor of the current argument elementary tree
8:          for each node from the target constituent to the root of the parse tree do
9:             take the node as the root of the current argument elementary tree
10:            if the argument etree and the predicate etree are compatible (i.e. no such
               node that belongs to both the predicate etree and the argument etree, and it
               is a non-root node for both the predicate etree and the argument etree) then
11:               if the number of the intermediate elementary trees between the predicate
                  etree and the argument etree is less than 3 then
12:                  for each possible intermediate etree (if it exists. Similar way to generate
                     it as to generate the argument etree) that is directly connected to the
                     predicate etree do
13:                     denote it as intermediate etree1
14:                     for each possible intermediate etree (if it exists. Similar way to gener-
                        ate it as to generate the argument etree) that is directly connected to
                        the argument etree do
15:                        denote it as intermediate etree2
16:                        a new LTAG derivation tree is generated based on a distinctive com-
                           bination of the predicate etree, the argument etree, and the inter-
                           mediate etree1, and the intermediate etree2 (In the combination,
                           intermediate etree1 and intermediate etree2 are optional.)
17:                     end for
18:                  end for
19:               end if
20:            end if
21:         end for
22:      end for
23:   end for
24: end for
```

roles can be reliably distinguished by possibly different latent LTAG features, and thus the latent features can be used discriminatively using a max-margin SVM classifier that only picks the most predictive features. Thus, in this work we explore the use of Latent SVMs (LSVM), which is a relatively new methodology for discriminative training introduced by (Felzenszwalb, McAllester, and Ramanan, 2008) for object detection in computer vision. LSVMs are a generalization of classic SVMs to handle classification problems with latent structures with the properties we described above.

4.3.1 Introduction

Let $D = \{\langle x_i, y_i \rangle\}_1^n$ be a set of labeled examples where x_i is an input and y_i is its output label. In classical SVMs, a weight vector $\mathbf{w}$ needs to be trained from D by optimizing the following objective function,

$$\arg\min_{\mathbf{w}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \max(0, 1 - y_i \mathbf{w} \cdot \Phi(x_i)) \tag{4.1}$$

where Φ is a feature vector for representing the input x, and the second term is the hinge loss. Each misclassification (when $y_i \mathbf{w} \cdot \Phi(x_i) < 1$) incurs a penalty proportional to the degree to which it violates the margin constraint.

In LSVM, we introduce an extra space of latent structures, and classify input instances while simultaneously searching for a good latent structure. The score assigned to an input x is then defined as,

$$f_{\mathbf{w}}(x) = \max_{h \in H(x)} \mathbf{w} \cdot \Phi(x, h) \tag{4.2}$$

where H is a set of latent structures. As in classical SVMs, the weight vector is trained by solving an analogous optimization problem,

$$\arg\min_{\mathbf{w}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \max(0, 1 - y_i f_{\mathbf{w}}(x_i)) \tag{4.3}$$

The LSVM framework allows us to enrich input instances with those latent structures which are most beneficial for the target task. Those latent annotations are preferred which induce low training error (the second term above), and at the same time, lead to a low

complexity classifier (the first term above). In other words, the training instances are embedded into an appropriate extended feature space specific to the given classfication task. If there is only one possible latent structure for each input ($|H(x)| = 1$), then LSVM reduces to classical SVM.

4.3.2 Semi-convexity

> ...in fact, the great watershed in optimization isn't between linearity and non-linearity, but convexity and nonconvexity.
>
> - R. Tyrrell Rockafellar, in SIAM Review, 1993

Convexity is an important property that allows even large sized optimization problems to be solved efficiently and reliably (Boyd and Vandenberghe, 2004). SVMs have been formulated as a convex optimization problem and have been widely used to solve many applications with a large data setting (Cristianini and Shawe-Taylor, 2000). When we generalize SVMs to LSVMs, we wish to preserve convexity. A proof for the convexity of objective function in LSVM is described in (Felzenszwalb, McAllester, and Ramanan, 2008) as follows:

"Note that $f_w(x)$ in equation (4.2) is a maximum of functions each of which is linear in w. Hence $f_w(x)$ is convex in w. This implies that the hinge loss $\max(0, 1 - y_i f_w(x_i))$ is convex in w when $y_i = -1$. This is, the loss function is convex in w for negative examples. We call this property of the loss function *semi-convexity*."

"Consider an LSVM where the latent domains $H(x_i)$ for the positive examples is restricted to a single choice. The loss due to each positive example is now convex. Combined with the semi-convexity property, equation (4.3) becomes convex in w."

In fact, when the set of latent structures $H(x_i)$ is restricted to a single element set, then LSVM becomes a regular SVM for which the convexity is satisfied.

4.3.3 Implementation: Coordinate Descent Algorithm

The description above implies a coordinate descent algorithm, alternating between optimizing h for positive examples, and optimizing w. The search is guaranteed to arrive at a local optimum in equation (4.3). Optimizing w with fixed structure for only the positive

examples are not entirely straightforward, so Felzensawalb et al. (2008) implemented an approximation, which fixes h for all examples. This results in an alternative coordinate descent, iterating over the following two steps until convergence:

1. Fix w, find the max structure for each training example:

$$\forall i : h_i \leftarrow \arg \max_{h \in H(x_i)} w \cdot \Phi(x_i, h)$$

2. Using features derived from $h_{[1...n]}$, find a new parameter w, which minimizes the standard SVM classification objective

$$\lambda \|w\|^2 + \sum_{i=1}^{n} \max(0, 1 - y_i \cdot w \cdot \Phi(x_i, h))$$

The above algorithm can be easily implemented, using the current weights to find the max latent structure h, turning the structure into labeled feature vectors, and training a classic SVM to classify the vectors. The resulting parameters w provide weights for the next round of training.

In our case the space of latent features are extracted from a structured source, namely each latent LTAG derivation tree. However, the derivation tree is mapped into an unstructured set of latent features that are used in a linear kernel within the LSVM framework. More structured latent variables can also be used in LSVMs which need more sophisticated training algorithms (Yu and Joachims, 2009; Wang and Mori, 2009).

4.3.4 Previous Work

As mentioned, LSVM has been successfully applied to the task of object detection in images (Felzenszwalb, McAllester, and Ramanan, 2008).

Similar idea was adopted for NLP in (Cherry and Quirk, 2008). The primary goal of their work was to construct a language model by training a classifier to judge the input sentences. In (Cherry and Quirk, 2008), parse trees are taken as latent structures for the input sentences and a parser-classifier is trained with an alternative objective function so that "good" sentences receive high-scoring trees, while "bad" sentences do not. The sign of the score will be used as a class label to reflect the sentence quality. This method has been shown in (Cherry and Quirk, 2008) to significantly outperform tri-gram based models.

Our work pioneers the application of LSVM to the SRL task. Following are the details of this application.

4.4 Semantic Role Labeling with LSVM

We construct a semantic role labeling system by using a LSVM where the hidden features are extracted from multiple LTAG derivation trees. Our model considers the LTAG derivation trees as latent structures underlying the Penn Treebank (PTB) syntactic parse trees and tries to learn predictive features that exist in these latent structures for SRL instances. Multiple derivation trees can provide an enriched feature space for the SRL task; if learned properly, the LTAG features from various trees are expected to improve SRL performance.

4.4.1 SRL Instance Generation

We use the commonly used architecture (as shown in Figure 1.1) for building our SRL system. It consists of four stages: **pruning**, **argument identification**, **argument classification** and finally optionally we perform a **multi-class classification** by breaking ties between individual argument classifiers based on their prediction score. Following the same strategy as described in Chapter 3, in the pruning stage we only allow nodes that are sisters to nodes on the path from the predicate to the root *in the parse tree*. We then generate the SRL training instances (positive and negative) from the pruned PTB trees. As in most other SRL systems, this instance generation method makes an independence assumption among SRL instances extracted from the same sentence. The final chosen LTAG derivation may vary across SRL instances that are extracted from the same sentence. Table 4.1 shows the number of positive and negative instances that are generated from our training data for LSVM training.

4.4.2 LSVM Training

For SRL identification we train a binary classifier and for SRL classification we use the standard "one-vs-all" scheme. For each given training point $\langle x_i, y_i \rangle$, x_i is a constituent from the derived tree, and y_i is the output label. For identification, y indicates whether the current constituent that dominates a span is an argument or not. For A_k classification

Task	Positive Examples	Negative Examples
ID	219,556	349,629
A0	60,351	159,205
A1	80,486	139,070
A2	17,191	202,365
A3	3,005	218,551
A4	2,461	217,095
AM	55,994	163,562

Table 4.1: The number of positive and negative training examples for the identification and classification tasks.

task, y indicates whether the current constituent, which has been already identified as an argument, has the A_k semantic role.

For LSVM training, we repeatedly alternate between the following steps mentioned in Section 4.3:

- Updating the latent derivation trees based on the current weights, and
- Learning the weights using a standard SVM training method based on the updated latent annotations.

For SVM training in the second step, we use the freely available SvmSgd package[5]. It is a straightforward implementation of *stochastic gradient descent* (SGD) which is an online learning algorithm for SVMs. The SGD algorithm selects a random training example at time t, and perform the following weight updates:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla F^{(t)}(\mathbf{w}^{(t)})$$

where $F^{(t)}(\mathbf{w})$ is the objective function specialized for the example chosen at time t, and η is the learning rate. The gradient $\nabla F^{(t)}(\mathbf{w})$ for our case is

$$\begin{cases} \lambda \mathbf{w} - y^{(t)} \Phi(x^{(t)}, h^{(t)}) & \text{if } y^{(t)} \mathbf{w} \cdot \Phi(x^{(t)}, h^{(t)}) < 1 \\ \lambda \mathbf{w} & \text{otherwise} \end{cases}$$

We use online learning for two reasons. Firstly, we have a very large dataset (the number of sentences in the PropBank times the number of predicates times the number of candidate

[5] `http://leon.bottou.org/projects/sgd`

nodes per predicate). We also have about 1.2 million features that can be potentially active at any given time. Thus a full quadratic programming search is expensive even for one run of SVM training. Secondly in cases of large datasets with sparse features, online learning can outperform batch learning (Liang and Klein, 2009). Our experience was similar: in preliminary experiments when using SVM-light (Joachims, 1999) for training the SVM in each iteration of our LSVM method, the performance was degraded compared to the use of SGD for training the same SVM.

To be more specific, in contrast to batch learning, where gradients are accumulated over all samples of the data set for the same weight setting before a gradient step is performed, a tiny gradient step is done here after each observations has been processed. For every epoch[6] the order in which the samples of the training data set are processed is randomized. A detailed discussion of the advantages of stochastic gradient descent over batch learning can be found in (Bottou and LeCun, 2003). More empirical as well as theoretical results concerning the stochastic gradient descent algorithm can also be found in (Bottou and LeCun, 2003).

4.4.3 Features

Table 4.2 lists the features we proposed that are extracted from the LTAG derivations, which overlap significantly with the ones used in Chapter 3 but we add n-gram features on etree spines. Our LSVM based SRL system (LSVM-SRL) uses features from both Table 3.1 and Table 4.2.

[6]One complete pass of the algorithm through all training samples is called epoch.

Table 4.2: Features from latent LTAG derivation trees used in our system. Head related features in Table 3.1 will be determined by the current LTAG derivation tree when they are used with the features in this table.

• pred etree (plus POS → voice)	
• arg etree (plus POS → NE if any)	
• reln of arg etree to pred etree: {child, parent, sibling, grandparent, uncle, other}	
• distance between arg etree and pred etree: {0, 1, 2}	
• label/index of attachment node between pred etree and parent (index of root = 0)	[1]
• position1 of pred etree to its parent etree	
• intermediate etree #1: predicate's parent etree, POS and lemma of anchor	[2]
• label/index of attachment point of arg etree and its parent OR	[3]
• label/index of attachment point of arg etree and its child	[4]
• intermediate etree #2: arg's parent etree, POS and lemma of anchor OR	[5]
• arg's child etree, POS and lemma of anchor	[4]
• position2 of arg etree to its parent	
• label of attachment between pred's parent etree and arg's parent etree	[6]
• position3 of pred's parent etree to arg's parent etree	
• position1 + position2 + position3 + relation	
• position1 + position2 + position3 + distance	
• relation + distance	
• upto 9 trigrams of arg etree spine (from the head)	
• upto 9 trigrams of pred etree spine (from the pred)	

when relation is:

[1] parent or sibling or grandparent or uncle.

[2] sibling or grandparent or uncle.

[3] child or sibling or uncle.

[4] grandparent.

[5] sibling or uncle.

[6] uncle.

4.5 Experiments

4.5.1 Experimental Setup

Similar to Chapter 3, training data (PropBank Sections 02-21), development set data (Section 24) and test data (Section 23) are taken from CoNLL-2005 shared task data[7] All the necessary annotation information such as predicates, parse trees[8] as well as named entity labels are part of the data. Our evaluation is done for identification and {A0, A1, A2, A3, A4, AM} using measures of *Precision/Recall/F-score*. We follow the same evaluation criteria as used in Chapter 3. For more details about evaluation please refer to Section 3.4.1 of Chapter 3.

4.5.2 SvmSgd Parameters

A proper adoption of learning rate is important to help the stochastic gradient descent converge faster. In SvmSgd, the learning rate has the form $1/\lambda(t + t_0)$ where

- λ is the regularization constant that has default value $1e - 4$ in the implementation;
- t is the index of the current example;
- the constant t_0 is determined heuristically to ensure that the initial update is comparable to the expected size of the weights. It has default value 0 in the implementation;

At each point in the training process, the weight update continues through a fixed number of epochs (experimentally set to 5 in the implementation). We run the package with all the above parameters for our experiments.

It must be mentioned is that the SvmSgd is implemented as a primal optimizer for a *linear* SVM, which indicates that the mapping from the the input space to the feature space is linear. In this sense, it is straightforward to obtain the updated weights after each training point is explored. Section 4.4.2 has given a detailed discussion.

[7] http://www.lsi.upc.es/∼srlconll/.

[8] from Charniak's parser (Charniak, 2000)

4.5.3 Baselines

We have built two baseline systems to compare the relative utility of the LSVM for SRL:

- **Baseline1** is an SRL system that uses the features in Table 3.1 only. No LTAG derivation is introduced for each SRL instance.
- **Baseline2** is an SRL system that uses the features in Table 3.1 and Table 4.2. The LTAG derivation that is introduced for each SRL instance is generated using *Magerman-Collins* head percolation rules and *sister-adjunction* operation. This baseline replicates the SRL method in Chapter 3.

For a fair comparison, we use SGD as the trainer for the two baseline systems.

4.5.4 Results

The initial setting of our LSVM-based SRL (LSVM-SRL) is the same as Baseline2: the weights trained from Baseline2 are the initial weights for LSVM training. Then we alternate between learning/updating the weights using SGD and selecting the best latent derivation based on the updated weights. Table 4.3 shows the *Precision/Recall/F-score* of LSVM-SRL and the baselines for each individual argument prediction. Compared to the two baselines, LSVM improves the SRL F-scores significantly in all cases. Even for A2, for which Baseline2 failed to show improvements, LSVM outperforms the purely phrase-structure based baseline (Baseline1). The precision obtained by each LSVM binary classifier for each argument type is near perfect.

Identification and Classification Results

Figure 4.5 shows the learning curve evaluated on test data across the iterations of LSVM training. The largest improvement happens in the iteration 1 where the weights are updated for the first time; the performance levels off after a few iterations. A similar trend has been observed in the development set.

Table 4.4 shows the results of LSVM-SRL using the CoNLL-2005 evaluation program on the CoNLL test data for the given set of argument types. We compare our scores for *A0, A1, A2, A3, A4* with the system that has the best reported results on the CoNLL-2005

class	Baseline1 (p/r/f)%			Baseline2 (p/r/f)%			LSVM-SRL (p/r/f)%			stopping point
ID	87.71	84.86	86.26	89.00	85.21	87.01	98.96	86.38	**92.74**	12
A0	86.46	85.30	85.87	87.87	87.50	87.68	99.26	90.31	**94.57**	18
A1	78.70	83.72	81.13	84.56	82.16	83.34	99.69	82.00	**89.99**	22
A2	85.04	73.91	79.09	83.00	74.86	78.72	99.26	73.35	**84.36**	26
A3	77.04	65.82	71.00	83.46	67.09	74.39	98.48	76.47	**86.09**	18
A4	77.42	79.12	78.26	90.14	70.33	79.01	98.77	79.21	**87.91**	20
AM	80.85	81.39	81.11	82.10	81.87	81.99	97.73	85.31	**91.10**	22

Table 4.3: Results for each individual binary classifier comparing Latent SVM with Baseline1, Baseline2 for the argument identification and classification tasks. The stopping point was determined based on accuracy on the development set.

shared task dataset (Toutanova, Haghighi, and Manning, 2008). We also compare with the best reported scores from various systems on each argument type. For *A0, A1, A2, A3*, LSVM-SRL gives the best precision/recall/f-scores by a large margin. However, for A4, it reports the lowest scores. One possible reason is that the prediction of A4 is not as confident compared to classifiers for other argument types.

We also evaluated our LSVM-SRL on the multi-class SRL classification task, where the entire sequence of predictions for the given argument set is examined. In terms of the argument set, we compare against (Toutanova, Haghighi, and Manning, 2008)[9] who obtain an f-score of 82.5% while we obtain (p/r/f%): 94.86/82.39/88.2.

Label	Our system (p/r/f)%			Toutanova et al. (2008)			CoNLL-2005 best scores per label			
A0	**98.77**	**92.45**	**95.51**	88.37	88.91	88.64	88.32	88.30	88.31	(Haghighi et al., 2005)
A1	**92.00**	**79.09**	**85.06**	81.50	81.27	81.38	82.25	77.69	79.91	(Punyakanok et al., 2005)
A2	**98.91**	**73.24**	**84.16**	73.44	68.74	71.01	72.55	68.11	70.26	(Haghighi et al., 2005)
A3	**64.65**	**73.99**	**69.00**	75.00	55.49	63.79	76.38	56.07	64.67	(Màrquez et al., 2005)
A4	25.81	78.43	38.83	74.74	69.61	72.08	**83.91**	**71.57**	**77.25**	(Punyakanok et al., 2005)

Table 4.4: Comparison (Precision/Recall/F-score%) of LSVM-SRL, the system having the best performance on the CoNLL-2005 shared task (Toutanova, Haghighi, and Manning, 2008), and various other papers that report the best score for each individual label. To the best of our knowledge these are the state of the art accuracies for individual labels on this dataset.

[9]There are some caveats to this comparison. They use PropBank data and top-10 parse trees from Charniak parser (2000) and we use CoNLL 2005 data. They predict *A5* arguments while we do not, but there are only 5 instances in the test set.

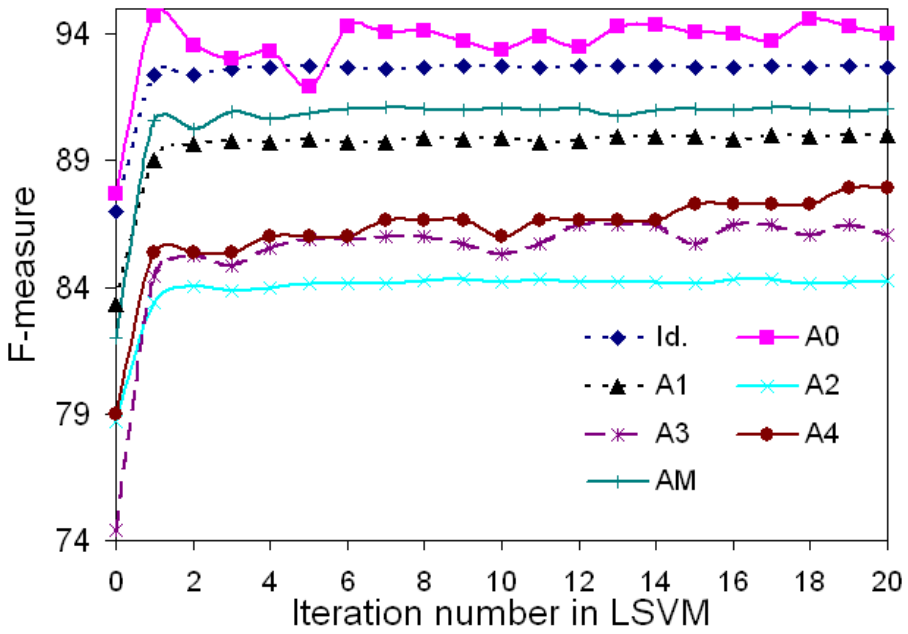

Figure 4.5: F-score of identification and classification tasks on test data for each iteration of LSVM.

4.5.5 Analysis

To better understand the behavior of LSVMs, we analyse various aspects of the LSVM-SRL algorithm and experimental results.

In each training iteration LSVM picks the argmax latent LTAG derivation. Are the latent derivation trees stabilized during the LSVM learning iterations? In each group of the two bars in Figure 4.6, the first bar shows the percentage of the instances for which the argmax derivation tree has not been observed in *all* previous iterations, and the second bar plots the percentage of the instances in the training data for which the latent derivation tree is new compared to the immediately previous iteration. The model converges pretty quickly to its best latent annotation for the training data. For over 90% of the instances the derivation trees are changed in the first iteration, which is a drastic departure from the initial derivation trees (from Baseline2). The tendency to pick unseen derivation trees exists throughout the learning process, although it drops significantly after a few iterations.

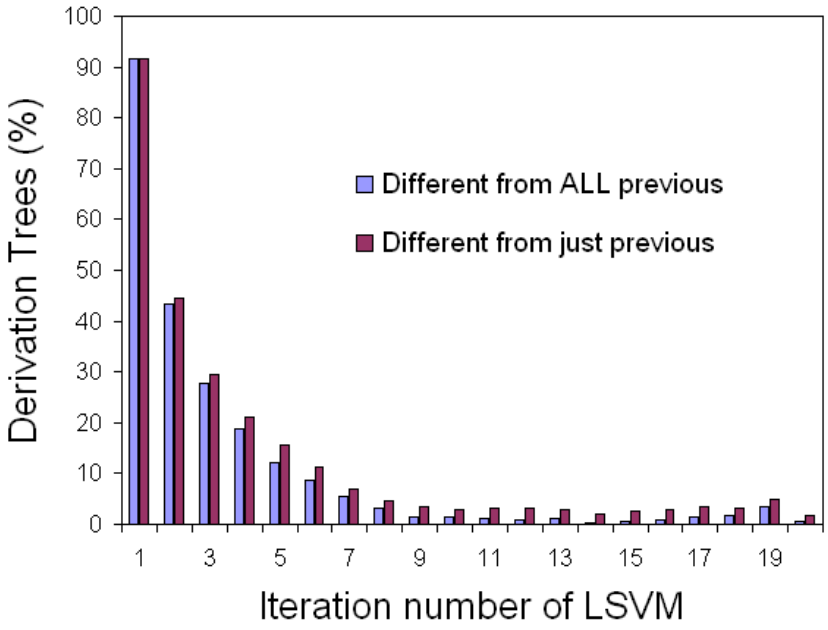

Figure 4.6: For each LSVM iteration in the identification task, the percentage of the training examples that have derivation trees that are different from the previous iteration and the percentage of the training examples that have derivation trees that have not been observed in all previous iterations.

Does LSVM training effectively minimize the complexity of the learned model by capturing feature sparsity? Each newly-selected previously unseen derivation tree (in Figure 4.6) brings a set of new active features[10] into the model. Although new active features are added constantly to the model, Figure 4.7 shows that the total number of active features decreases drastically during the first few learning iterations, and then stabilizes. In fact the model has picked a small set of features (8K features from the pool of 1,242,869 all possible features) having strong predictions for each individual task. Figure 4.8 shows the average entropy of conditional probability of labels given features for the A1 classification task. The reduction of average entropy confirms that LSVM picks highly discriminative features. We have observed similar trend for other classification tasks, except the identification task for which the combination of several features seems to be necessary for making a good prediction.

[10] Active features are those which have non-zero values in the learned weight vector.

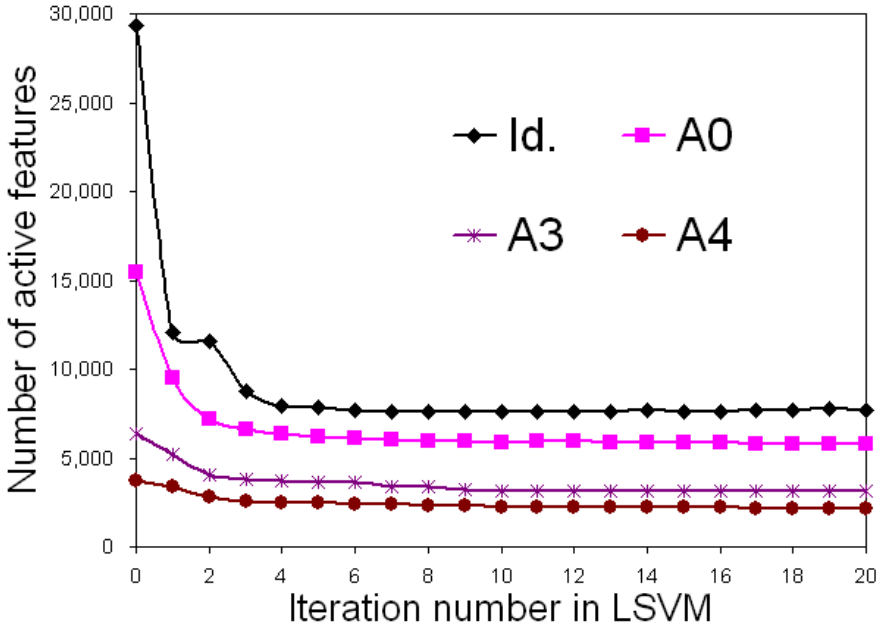

Figure 4.7: Number of active features during the iteration of LSVM for different tasks. These curves for the other classification tasks are similar to that of A0.

Does the initialization of the weight vector in LSVM matter? In our main experiments we initialize the weights based on SVM training using linguistically plausible Magerman-Collins head rules. To compare against this initialization, for each training instance, we randomly select a derivation tree from its space of latent derivation trees, and train the LSVM from this random initial condition. The results show that proper initialization matters a lot. For example, in the A0 classification task, we have observed that p/r/f% is $84.54_{\pm 8.00}/86.44_{\pm 2.12}/85.33_{\pm 4.49}$ over five independent runs. This is far better than that of random initial weights $71.15_{\pm .79}/86.11_{\pm .29}/77.92_{\pm .36}$ (mainly due to the precision), but less than the performance of our best system reported in Table 4.3.

4.5.6 Weight updates in Training Process

As we observed before, the selected derivation tree features tend consolidate towards a small stable set of features across LSVM iterations. In this part, we try to gain a quick intuition through the observation of weight updates across the training process. Due to the

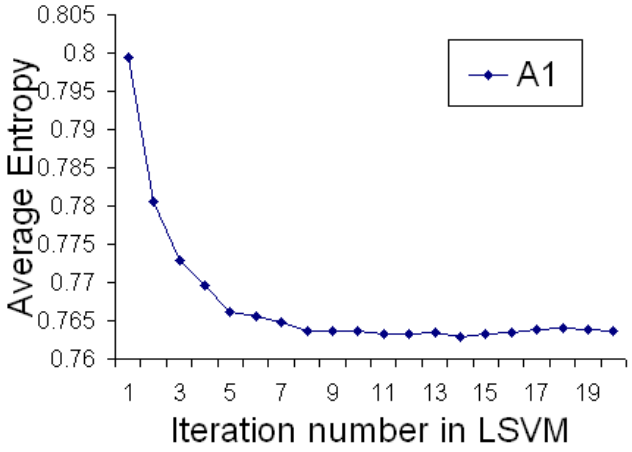

Figure 4.8: For each feature f, we compute the entropy of its predictive distribution $(\frac{C_f^+}{C_f^+ + C_f^-}, \frac{C_f^-}{C_f^+ + C_f^-})$ where C_f^+ / C_f^- are the frequencies of f in the positive / negative training instances. The y-axis is weighted average of the entropy over active features; where weights are proportional to the feature frequencies.

huge number of active features, it is infeasible to give an overview of the weight updates for all or even part of those features. Plus it is difficult to track the behavior of a linear model by looking at individual weights, since the final result is a dot product of individuals. However, we can get a first approximation of feature value by looking at the way that the weights change over the training iterations. Therefore, we start our observation by first picking an example whose label was initially incorrectly predicted before applying LSVM. See the example in Figure 4.9 where *do* is the predicate and *NP* that dominates *the program* is the candidate constituent; and where *the program* plays as semantic role "A1" for the predicate *do*. To avoid a busy figure we select 9 of them as representatives with the hope to find the smart updates on salient features. The selection of the representative features is based on the trend of their weight updates as the number of iterations increases. For example, in Figure 4.10, feature#1144[11] and feature#193 both have weight 0 in the beginning. With subsequent iterations, weight of feature#1144 increases while weight of feature #193 decreases. Similar trend is observed for feature#428 and feature#119, which

[11]The numbers we use here are just an easy shorthand for features in the feature pool.

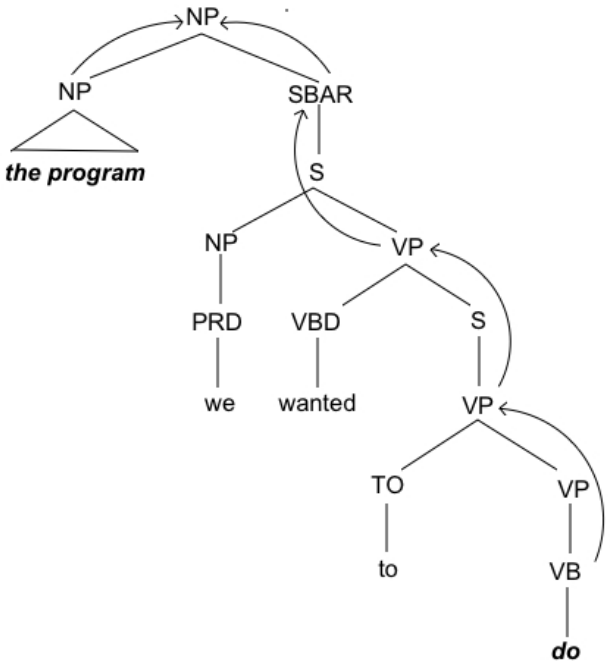

Figure 4.9: Tree fragment of a training example

both have non-zero initial weights. Feature#24 has an initial positive weight that becomes negative with subsequent iterations. The opposite is true for feature#181. For interest, we also select a tri-gram feature of predicate spine valued *NULL>NULL>NULL*. Feature#17, Feature#18, Feature#19 are such features. Figure 4.10 shows the updating process of these features over 50 iterations of training.

By looking up the features in the feature pool, we found that it is hard to tell if the feature that has an increasing weight is more salient than the feature that has a decreasing weight. For example, feature#1144 has an increasing weight, especially in the first 20 iterations. The feature is *VB>VP>VP*, one of the tri-grams of the current predicate spine. Feature#181 denotes the voice of the current predicate, which is 0 in this case. Feature#119 is the LCA (refer to Table 3.1) feature which is *NP>NP*. Feature#17, that has value *NULL>NULL>NULL* (one of the tri-grams of the predicate spine), has also experienced dramatic weight updates, especially in the first few rounds.

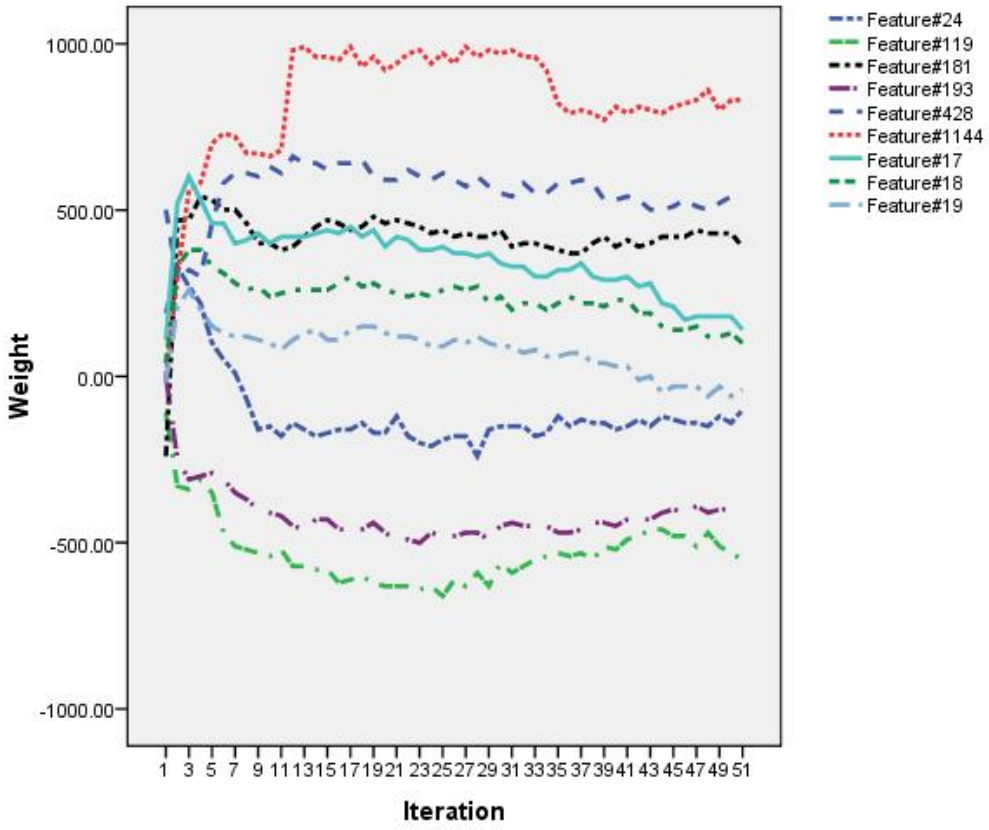

Figure 4.10: Snapshot of weight updating process across 50 iterations for 9 sample features from a training example.

Based on the above observation, it is hard to give a specific example to illustrate how well the LSVM method works for feature weight updates. However, we would say that the method works for weight updating in that the linear combination of the updated weights, if not a single updated weight, gives a great improvement to the system performance.

4.6 Conclusion and Future Work

In this chapter, we generalized the work in Chapter 3 and proposed to apply a novel learning framework, latent support vector machines (LSVMs), to the SRL task with multiple LTAG derivation trees. First we treated the LTAG derivation trees as latent variables, through

which we provided an enriched feature space for the SRL task. To solve the SRL task with latent structures, we adopted LSVM for each binary classification scenario and an overall improvement was obtained. We recovered a huge number of latent LTAG derivation trees by enumerating the combinations of predicate elementary trees, argument elementary trees and intermediate elementary trees. We used SvmSgd package, a straightforward implementation of online gradient descent algorithm, as the training tool in each LSVM iteration. The method has shown great advantage in improving the system performance over the two baseline systems.

We presented the experimental results and analysis on the results. In addition, we used an example to observe how the feature vectors are updated over the LSVM iterations in order to find the salient features introduced by the latent structures.

As far as the related work is concerned, (Vickrey and Koller, 2008) is the only other work (that we know of) that uses latent features for SRL. They use hand-written sentence simplification rules and search for the most useful set of simplifications that can improve training error based on a log-linear model. Their model sums over the set of possible simplifications in contrast to our LSVM approach. In terms of SRL results, we outperform them by a big margin on the CoNLL 2005 dataset. In future work, we would like to explore our approach in the probabilistic setting using log-linear models and marginalizing out the latent variables.

We believe that our success in applying LSVM to SRL has provided useful evidence to the NLP research community for the general applicability of LSVM to complex and diverse NLP tasks.

Chapter 5

SRL for Biomedical Relationship Extraction

In this chapter, we present our work of applying SRL related features to a real-world information retrieval task. Specifically, we proposed a ternary relationship extraction method primarily based on the rich syntactic information that has been used for the SRL task. We extract PROTEIN-ORGANISM-LOCATION relations from the text of biomedical articles to facilitate the biologists to generate the annotated corpus. Different kernel functions are used with an SVM learner to integrate two sources of information from syntactic parse trees: (i) specific syntactic features extracted along the path between entities, and (ii) features from entire trees using a tree kernel. We use the large number of syntactic features that have been shown to be useful for Semantic Role Labeling (SRL) and apply them to the relation extraction task. Our experiments show that the use of rich syntactic features significantly improves performance over the system using only shallow word-based features.

5.1 Introduction

A biomedical functional relation (biomedical relation for short) states interactions among biomedical substances. For instance, the PROTEIN-ORGANISM-LOCATION (POL) relation that we study in this work presents information about where a PROTEIN is located in an ORGANISM, thus providing valuable clue to the biological function of the PROTEIN

and helping to identify suitable drug, vaccine and diagnostic targets. Figure 5.1 illustrates possible locations of proteins in Gram+ and Gram-, two straints of bacterium.

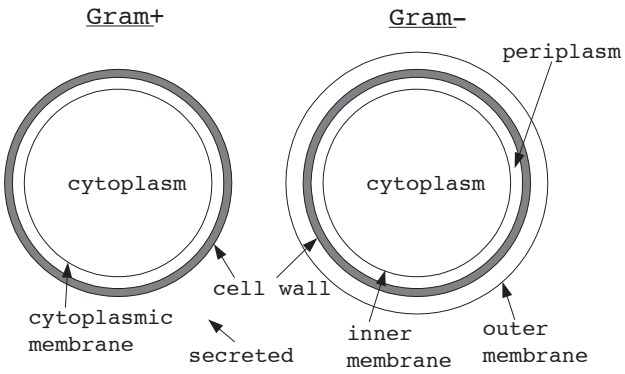

Figure 5.1: Illustration of bacterial locations

Previous work in biomedical relation extraction task (Sekimizu, Park, and Tsujii, 1998; Blaschke et al., 1999; Feldman et al., 2002) suggested the use of predicate-argument structure by taking verbs as the center of relation expression. In contrast, in this work we directly link protein named entities to their locations; an approach was proposed in (Claudio, Lavelli, and Romano, 2006) that solely considers the shallow semantic features extracted from sentences.

For relation extraction in the newswire domain, syntactic features have been used in a generative model (Miller et al., 2000) and in a discriminative log-linear model (Kambhatla, 2004). In comparison, we use a much larger set of syntactic features extracted from parse trees, many of which have been shown useful in semantic role labeling. Also, kernel-based methods have been used for relation extraction (Zelenko, Aone, and Richardella, 2003; Culotta and Sorensen, 2004; Bunescu and Mooney, 2005) on various syntactic representations, such as dependency trees or constituency-based parse trees. In contrast we explore a much wider variety of syntactic features in this work. To benefit from both views, a composite kernel (Zhang et al., 2006) integrates the flat features from entities and structured features from parse trees. In our work, we also combine a linear kernel with a tree kernel for improved performance.

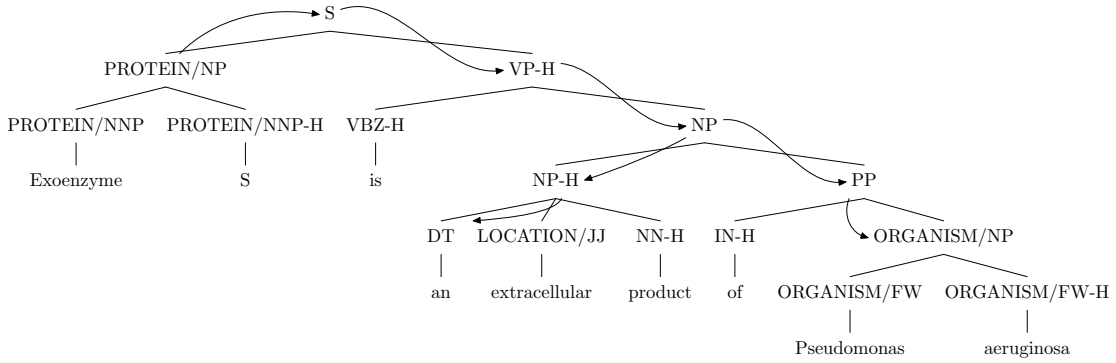

Figure 5.2: An example of POL ternary relation in a parse tree

5.2 SRL Features for Information Extraction

Figure 5.2 shows one example illustrating the ternary relation we are identifying. In this example, "Exoenzyme S" is a PROTEIN name, "extracellular" a LOCATION name and "Pseudomonas aeruginosa" an ORGANISM name. Our task is to determine whether there exists a "PROTEIN-ORGANISM-LOCATION" relation among these three named entities (NEs).

To simplify the problem, we first reduce the POL ternary relation extraction problem into two binary relation extraction problems. Specifically, we split the POL ternary relation into binary relation as follows:

- PO: PROTEIN and ORGANISM
- PL: PROTEIN and LOCATION

Notice that the ORGANISM-LOCATION relation is ignored because it is irrelevant to PROTEIN and less meaningful than PO and PL relations. Based on this simplification, and following the idea of semantic role labeling, we take the PROTEIN name in the role of the predicate (verb) and the ORGANISM/LOCATION name as its argument candidates in question. Then the problem of identifying the binary relations of PO and PL has been reduced to the problem of argument classification problem given the predicate and the argument candidates. The reason that we pick PROTEIN names as predicates is that we

assume PROTEIN names play a more central role in linking the binary relations to the final ternary relations.

Compared to a corpus for the standard SRL task, there are some differences: first is the relative position of PROTEIN names and ORGANISM/LOCATION names. Unlike the case in SRL, where arguments locate either before or after the predicate, in this application it is possible that one named entity is embedded in another. A second difference is that a predicate in SRL scenario typically consists of only one word; here a PROTEIN name can contain up to 8 words.

Note that we do not use PropBank data in our model at all. All of our training data and test data is annotated by domain expert biologists and parsed by Charniak-Johnson's parser (released in 2006). When there is a misalignment between the named entity and the constituent in the parse tree, we insert a new NP parent node for the named entity.

5.3 System Description

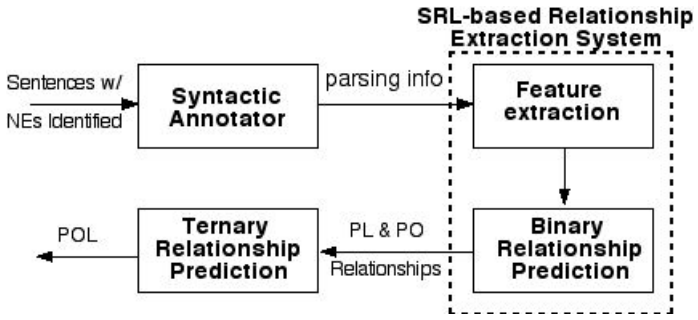

Figure 5.3: High-level system architecture

Figure 5.3 shows the system overview. The input to our system consists of titles and abstracts that are extracted from MEDLINE[1] records. These extracted sentences have been annotated with the NE information (PROTEIN, ORGANISM and LOCATION). Syntactic Annotator inserts the head information to the parse trees by using the Magerman-Collins head percolation rules. The main component of the system is our SRL-based

[1] http://medline.cos.com/

• each word and its Part-of-Speech (POS) tag of the PRO name
• head word (hw) and its POS of the PRO name
• subcategorization that records the immediate structure that expands from the PRO name. Non-PRO daughters will be eliminated
• POS of parent node of the PRO name
• hw and its POS of the parent node of the PRO name
• each word and its POS of the ORG name (in the case of "PO " relation extraction).
• hw and its POS of the ORG name
• POS of parent node of the ORG name
• hw and its POS of the parent node of the ORG name
• POS of the word immediately before/after the ORG name
• punctuation immediately before/after the ORG name
• feature combinations: hw of PRO name_hw of ORG name, hw of PRO name_POS of hw of ORG name, POS of hw of PRO name_POS of hw of ORG name
• path from PRO name to ORG name and the length of the path
• trigrams of the path. We consider up to 9 trigrams
• lowest common ancestor node of the PRO name and the ORG name along the path
• LCA (Lowest Common Ancestor) path that is from the ORG name to its lowest common ancestor with PRO name
• relative position of PRO name and ORG name. In parse trees, we consider 4 types of positions that ORGs are relative to PROs: before, after, inside, other

Table 5.1: Features adopted from the SRL task. PRO: PROTEIN; ORG: ORGANISM

relation extraction module, where we first manually extract features along the path from the PROTEIN name to the ORGANISM/LOCATION name and then train a binary SVM classifier for the binary relation extraction. Finally, we fuse the extracted binary relations to the ternary relations and obtain the final results. In contrast with this discriminative model, a statistical parsing based generative model (Shi, Sarkar, and Popowich, 2007) has been built up for the same task where the named-entities and their relations are extracted together. Since our final goal is to facilitate the biologists to generate the annotated corpus, we plan to take the annotated named entities from the generative model as our input.

As a discriminative feature-based relation extraction system, identification of important features is crucial to the task. Table 5.1 and Table 5.2 listed the features that are used in the system.

• subcategorization that records the immediate structure that expands from the ORG name. Non-ORG daughters will be eliminated
• if there is an VP node along the path as ancestor of the PRO name
• if there is an VP node as sibling of the PRO name
• if there is an VP node along the path as ancestor of the ORG name
• if there is an VP node as sibling of the ORG name
• path from PRO name to LCA and the path length (L1)
• path from ORG name to LCA and the path length (L2)
• combination of L1 and L2
• sibling relation of PRO and ORG
• distance between PRO name and ORG name in the sentence. (3 valued: 0 if nw (number of words) = 0; 1 if 0 < nw <= 5; 2 if nw > 5)
• combination of distance and sibling relation

Table 5.2: New features used in the SRL-based relation extraction system.

5.4 Experiments and Evaluation

5.4.1 Data Set

Our work is part of collaboration with molecular biologist to build up a POL named entity/relation annotated corpus from scratch. As in a real scenario, our experimental data set is derived from a small expert-curated corpus, in which the POL relations and relevant PROTEIN, ORGANISM and LOCATION NEs are labeled. It contains around 150k words, 565 relation instances for POL, 371 for PO and 431 for PL. The small size of our dataset is a common problem of many other bio-medical information extraction tasks[2].

5.4.2 Systems and Experimental Results

We built several models to compare the relative utility of the various types of rich syntactic features we can exploit for relation extraction. For various representations, such as feature vectors, trees and their combinations, we applied different kernels in a Support Vector Machine (SVM) learner. Specifically, we use Joachims' SVM-light[3] with default linear

[2]To ensure significance of our results, we do 5-fold cross validation in all our experiments.

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

Method	PL				PO				POL			
Measure	**P**	**R**	**F**	**A**	**P**	**R**	**F**	**A**	**P**	**R**	**F**	**A**
Baseline1	98.1	61.0	75.3	60.6	88.4	59.7	71.3	58.5	*57.1*	90.9	70.1	*56.3*
Baseline2	61.9	100.0	76.5	61.9	48.8	100.0	65.6	48.9	*59.8*	100.0	74.8	*59.8*
PAK	71.0	71.0	71.0	64.6	69.0	66.7	67.8	61.8	*66.0*	69.9	67.9	*62.6*
SRL	72.9	77.1	74.9	70.3	66.0	71.0	68.4	64.5	*70.6*	67.5	69.0	*65.8*
TRK	69.8	81.6	75.3	72.0	64.2	84.1	72.8	72.0	***79.6***	66.2	72.3	*71.3*
TRK+SRL	74.9	79.4	77.1	72.8	73.9	78.1	75.9	72.6	*75.3*	74.5	74.9	***71.8***

Table 5.3: Percent scores of Precision/Recall/F-score/Accuracy for identifying PL, PO and POL relations.

kernel to feature vectors and Moschetti's SVM-light-TK-1.2[4] with the default tree kernel. The models are:

Baseline1 is a purely word-based system, where the features consist of the unigrams and bigrams between the PROTEIN name and the ORGANISM/LOCATION names inclusively.

Baseline2 is a naive approach that assumes that any example containing PROTEIN, LOCATION names has the PL relation. The same assumption is made for PO and POL relations.

PAK system uses predicate-argument structure kernel (PAK) based method. PAK was defined in (Moschitti, 2004) and only considers the path from the *predicate* to the *target argument*, which in our setting is the path from the PROTEIN to the ORGANISM or LOCATION names.[5]

SRL is an SRL system which is adapted to use our new feature set. A default linear kernel is applied with SVM learning.

TRK system is similar to the PAK system except that the input trees are entire parse trees instead of PAK paths.

TRK+SRL combines full parse trees and manually extracted features and uses the kernel combination.

[4] `http://ai-nlp.info.uniroma2.it/moschitti/TK1.2-software/Tree-Kernel.htm`

[5] We also experimented with **PAK+SRL** but since the results were similar to using only **SRL**, we do not discuss it here.

5.4.3 Fusion of Binary relations

We predict the POL ternary relation by fusing PL and PO binary relations if they belong to the same sentence and have the same PROTEIN NE. The prediction is made by the sum of confidence scores (produced by the SVM) of the PL and PO relations. This is similar to the postprocessing step in SRL task that the semantic roles assigned to the arguments have to realize a legal final semantic frame for the given predicate.

5.4.4 Discussion

Table 5.3 shows the results we obtained when running on our data set with 5-fold cross validation. We evaluated the system performance for ternary relation extraction as well as the extraction of the two binary relations. The total accuracy of finding the correct ternary relation in the test data using rich syntactic features is 71.8% and we can see that it significantly outperforms shallow word-based features which obtains 56.3% accuracy[6].

By comparing the **PAK** model and **SRL** model, we observe that with the same path, a system based on manually extracted features boosts precision and accuracy and therefore obtains an overall better performance than the **PAK** model. In contrast to the SRL model for both PL and PO relation extraction, the **TRK** model obtains lower precision but higher recall. This means that the sub-structures considered by **TRK** are good in discriminating the instances but may not suffice. The addition of SRL features boosts the overall performance of **TRK** system to the best overall F-score by moderating precision and recall.

The gaps between models reinforce the fact that the path between named entities in the parse tree is very important for the relation extraction task. In particular, it illustrates that along this path, SRL-based syntactic features are discriminative as well as necessary for this task. In addition, our experiments showed that some features outside this path can contribute to the task to some extent.

[6]We highlight precision and accuracy of finding the ternary relations in text, i.e. distinguishing between positive and negative examples of relations as these are the most important figures for the domain expert biologists.

Chapter 6

Conclusions and Future Work

Semantic role labeling (SRL) has been attracting much attention in the research community as an important intermediate step in many important NLP applications. From the information extraction point of view, semantic role labeling can also be taken as an instance of "multi-way relationships" extraction problem where the relation between the given predicate and each of its arguments has to be identified and labeled. Lexicalized Tree Adjoining Grammar (LTAG), with its property of Extended Domain of Locality (EDL), is particularly suited to the SRL task. Initially motivated by the connection between SRL and LTAG, we started the effort of applying LTAG to the SRL task.

Previous work had shown the limitation of applying LTAG elementary trees to SRL that are extracted from using the standard way. In our work, we first applied the LTAG derivation trees that are produced by the immediately available LTAG parser to the SRL task. We also took advantage of the flexibility of defining LTAG elementary trees and mainly used the sister-adjunction operation to combine the generated LTAG elementary trees. The most challenging task of disambiguating complements and adjuncts in LTAG extraction has been bypassed and left to to be resolved by a probabilistic model of SRL. In this way the EDL property is used implicitly and could be made more favorable to SRL. Our experimental results have shown the positive impact of using LTAG features for the SRL task, and they were shown to be distinct and more accurate than tree kernels. As an extension, we treated multiple LTAG derivation trees as latent structures of a Penn Treebank tree and used latent support vector machines (LSVMs) for the SRL task with

the latent LTAG structures, which improves the SRL system performance significantly. As a summary, we have accomplished the following work in this research:

- We first explored a novel LTAG formalism – LTAG-spinal and its treebank for the SRL task and demonstrated the utility of this new resource for SRL. Our work has also contributed to the further development of LTAG-spinal resource.
- We proposed the use of LTAG derivation trees as an important additional source of features for the SRL task. These features, after combined with the standard features, can improve SRL system performance on a statistically significant level.
- As a generalization of the previous work, we recovered multiple LTAG derivations as latent structures of the given Penn Treebank parse tree and applied a novel learning framework - Latent Support Vector Machines (LSVMs) with a stochastic gradient descent (SGD) algorithm to SRL. This method significantly outperforms the state-of-the-art SRL systems. LSVMs are well suited for the SRL task: binary LSVM classifiers for core argument classification obtain near perfect precision.
- To implement the idea of treating SRL as one of the "multi-way relationships" extraction problems, we proposed and applied SRL framework and features for ternary relation extraction from the biomedical domain and obtained a better result superior to the use of word-level features.

In the future, we would like:

- To explore our approach to the current LSVM-based SRL system in the probabilistic setting using log-linear models and marginalizing out the latent variables.
- To explore the general application of LSVM to other NLP tasks. The latent SVM could potentially become a powerful tool for many NLP tasks. It applies whenever a hidden structure would be useful in a classification task.
- SRL can be viewed as a general relation extraction problem. We believe that the entire SRL framework (such as the architecture, and the features) can be more widely applied to various applications that involve relation extraction.

References

Abeillé, Anne and Owen Rambow, editors. 2001. *Tree adjoining grammars: formalisms, linguistic analysis and processing*. Center for the Study of Language and Information.

Abeillé, Anne, Yves Schabes, and Aravind K. Joshi. 1990. Using lexicalized tags for machine translation. In *Proceedings of the 13th Conference on Computational Linguistics*, pages 1–6, Morristown, NJ, USA. Association for Computational Linguistics.

Allen, James. 1994. *Natural Language Understanding*. Benjamin Cummings Publishing Company, New York.

Baker, C.F. and C.J. Fillmore. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING/ACL-1998)*.

Blaheta, Don and Eugene Charniak. 2000. Assigning function tags to parsed text. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2000)*, pages 234–240, Seattle.

Blaschke, C., M. Andrade, C. Ouzounis, and A. Valencia. 1999. Automatic extraction of biological information from scientific text: protein-protein interactions. In *Proceedings of AAAI Conference on Intelligent Systems in Microbiology (ISMB-1999)*, pages 60–77, Heidelberg, Germany, AAAI, Menlo Park, CA.

Bottou, Léon and Yann LeCun. 2003. Large scale online learning. In *Proceedings of the 17th Annual Conference on Neural Information Processing Systems (NIPS-2003)*, Vancouver and Whistler, British Columbia, Canada, December. MIT Press.

Boxwell, Stephen, Dennis Mehay, and Chris Brew. 2009. Brutus: a semantic role labeling system incorporating ccg, cfg, and dependency features. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 37–45, Suntec, Singapore, August. Association for Computational Linguistics.

Boxwell, Stephen A. and Michael White. 2008. Projecting propbank roles onto the ccgbank. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC-2008)*.

Boyd, Stephen and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press, March.

Braz, R., R. Girju, V. Punyakanok, D. Roth, and M. Sammons. 2005. An inference model for semantic entailment in natural language. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-2005)*.

Bunescu, R. C. and R. J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP-2005)*, pages 724–731, October.

Carreras, X. and L. Màrquez. 2004. Introduction to the CoNLL-2004 shared task. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL-2004)*, Boston, MA, USA.

Carreras, X. and L. Màrquez. 2005. Introduction to the CoNLL-2005 shared task. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL-2005)*, Ann Arbor, MI, USA, June.

Charniak, E. 2000. A maximum-entropy-inspired parser. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2000)*, Seattle.

Charniak, Eugene and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-2005)*, Ann Arbor, MI, USA, June.

Che, W., M. Zhang, T. Liu, and S. Li. 2006. A hybrid convolution tree kernel for semantic role labeling. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-2006)*, Sydney, Australia, July.

Chen, J. and O. Rambow. 2003. Use of deep linguistic features for the recognition and labeling of semantic arguments. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP-2003)*.

Chen, John. 2002. *Towards efficient statistical parsing using lexicalized grammatical information*. Ph.D. thesis, University of Delaware, Newark, DE, USA. Professor In Charge-Shanker, Vijay.

Chen, John and K. Vijay-Shanker. 2000. Automated extraction of TAGs from the Penn TreeBank. In *Proceedings of the 6th International Workshop on Parsing Technologies (IWPT-2000)*, Italy.

Cherry, Colin and Chris Quirk. 2008. Discriminative, syntactic language modeling through latent svms. In *Proceeding of Association for Machine Translation in the America (AMTA-2008)*.

Chiang, D. 2000. Statistical parsing with an automatically extracted tree adjoining grammars. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000)*, Hongkong, October.

Chiang, David and Daniel M. Bikel. 2002. Recovering latent information in treebanks. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 1–7, Morristown, NJ, USA. Association for Computational Linguistics.

Chou, W., R. Tsai, Y. Su, W. Ku;, T. Sung, and W. Hsu. 2006. A semi-automatic method for annotating a biomedical proposition bank. In *Proceedings of the Workshop on Frontiers in Linguistically Annotated Corpus -2006*.

Ciaramita, Massimiliano, Giuseppe Attardi, Felice Dell'Orletta, and Mihai Surdeanu. 2008.

Desrl: A linear-time semantic role labeling system. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL-2008)*, pages 159–177, Manchester, England, August. Coling 2008 Organizing Committee.

Claudio, G., A. Lavelli, and L. Romano. 2006. Exploiting shallow linguistic information for relation extraction from biomedical literature. In *Proceedings of the 11st Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006)*, Trento, Italy, April 5–7.

Cohn, Trevor and Philip Blunsom. 2005. Semantic role labelling with tree conditional random fields. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL-2005)*, Ann Arbor, MI, USA, June.

Collins, Michael. 1997. Three generative, lexicalized models for statistical parsing. In Philip R. Cohen and Wolfgang Wahlster, editors, *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL-1997)*, pages 16–23, Somerset, New Jersey.

Collins, Michael and James Brooks. 1995. Prepositional phrase attachment through a backed-off model. In *Proceedings of the Third Workshop on Very Large Corpora*, pages 27–38.

Cristianini, Nello and John Shawe-Taylor. 2000. *An Introduction to support vector machines and other kernel-based kearning methods.* Cambridge University Press, 1 edition, March.

Culotta, A. and J. Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-2004)*, pages 423–429, Barcelona, Spain, July.

Dang, H.T. and M. Palmer. 2005. The role of semantic roles in disambiguating verb senses. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-2005)*, Ann Arbor, MI, USA, June.

Deneefe, Steve and Kevin Knight. 2009. Synchronous tree adjoining machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-2009).*

Feldman, R., Y. Regev, M. Finkelstein-Landau, E. Hurvitz, and B. Kogan. 2002. Mining biomedical literature using information extraction. *Current Drug Discovery*, pages 19–23, October.

Felzenszwalb, Pedro F., David A. McAllester, and Deva Ramanan. 2008. A discriminatively trained, multiscale, deformable part model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR-2008).*

Fillmore, C.J., C. Wooters, and C.F. Baker. 2001. Building a large lexical databank which provides deep semantics. In *Proceedings of the 15th Paci Asia Conference on Language, Information and Computation (PACLIC15-2001)*, Hong Kong.

Ge, R. and R. J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, Ann Arbor, MI, USA, June.

Ge, R. and R. J. Mooney. 2006. Discriminative reranking for semantic parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (COLING/ACL-2006)*.

Gildea, D. and J. Hockenmaier. 2003. Identifying semantic roles using combinatory categorial grammar. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP-2003)*.

Gildea, D. and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 58(3):245–288.

Gildea, D. and M. Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*.

Giuglea, A.M. and A. Moschitti. 2006. Semantic role labeling via framenet, verbnet and propbank. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (COLING/ACL-2006)*.

Hacioglu, K., S. Pradhan, W. Ward, J. Martin, and D. Jurafsky. 2004. Semantic role labeling by tagging syntactic chunks. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL-2004)*.

Hacioglu, Kadri. 2004. Semantic role labeling using dependency trees. In *Proceedings of 20th International Conference on Computational Linguistics (COLING-2004)*, University of Geneva, Switzerland, August 23rd-27th.

Haghighi, Aria, Kristina Toutanova, and Christopher Manning. 2005. A joint model for semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 173–176, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Harabagiu, Sanda M., Cosmin A. Bejan, and Paul Morarescu. 2005. Shallow semantics for relation extraction. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-2005)*, Edinburgh, Scotland, UK.

Harbusch, Karin and Jens Woch. 2002. Integrated natural language generation with schema-tree adjoining grammars. In *Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing (CICLing 2002)*, pages 304–313, London, UK. Springer-Verlag.

Hindle, Donald. 1990. Noun classification from predicate-argument structures. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics (ACL-1990)*.

Jiang, Z. P. and H. T. Ng. 2006. Semantic role labeling of nombank: a maximum entropy approach. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP-2006)*.

Joachims, T. 1999. Making large-scale svm learning practical. *Advances in Kernel Methods - Support Vector Machines*.

Johansson, Richard and Pierre Nugues. 2008a. Dependency-based semantic role labeling of PropBank. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-2008)*, pages 69–78.

Johansson, Richard and Pierre Nugues. 2008b. The effect of syntactic representation on semantic role labeling. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING-2008)*, pages 393–400, Manchester, UK, August. Coling 2008 Organizing Committee.

Joshi, A. and Y. Schabes. 1997. Tree-adjoining grammars. *Handbook of Formal Languages*, 3.

Joshi, A. K. 1985. How much context-sensitivity is necessary for assiging structural descriptions: Tree adjoining grammars. *Natural Language Parsing*.

Joshi, Aravind K., L. S. Levy, and M. Takahashi. 1975. Tree adjunct grammars. *Journal of Computer and System Sciences*, 1(10).

Kambhatla, Nanda. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 22, Morristown, NJ, USA. Association for Computational Linguistics.

Kasper, Robert, Bernd Kiefer, and Klaus Netter. 1995. Compilation of hpsg to tag. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL-1995)*, pages 92–99, Cambridge, Massachusetts, USA, June. Association for Computational Linguistics.

Kate, R. and R. J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (COLING/ACL-2006)*.

Kipper, K., H.T.Dang, and M. Palmer. 2000. Class-based construction of a verb lexicon. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-2000)*.

Kipper, K., M. Palmer, and O. Rambow. 2002. Extending propbank with verbnet semantic predicates. In *Proceedings of the 5th Conference of the Association for Machine Translation in the Americas (AMTA-2002)*, Tiburon, CA, USA, October 6-12.

Kipper, K., B. Snyder, and M. Palmer. 2004. Extending a verb-lexcion using a semantically annotated corpus. In *Proceedings of the fourth International Conference on Language Resources and Evaluation (LREC-2004)*.

Levin, B. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. The University of Chicago Press.

Liang, P. and D. Klein. 2009. Online EM for unsupervised models. In *Proceedings of Human Language Technologies/The Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2009)*.

Lin, Chin-Yew and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-2003)*, pages 71–78, Morristown, NJ, USA. Association for Computational Linguistics.

Lin, Dekang and Patrick Pantel. 2001. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(4):343–360.

Litkowski, Ken. 2004. Senseval-3 task: word sense disambiguation of wordnet glosses. In Rada Mihalcea and Phil Edmonds, editors, *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 13–16, Barcelona, Spain, July. Association for Computational Linguistics.

Liu, Y. and A. Sarkar. 2006. Using LTAG-based features for semantic role labeling. In *Proceedings of the Eighth International Workhop on Tree Adjoining Grammar and Related Formalisms (TAG+8)*, Sydney, Australia, July 15–16.

Liu, Y. and A. Sarkar. 2007. Experimental evaluation of LTAG-based features for semantic role labeling. In *Proceedings of the 2007 Conference on Empirical Methods in Natural Language Processing (EMNLP-2007)*.

Loper, Edward, Szu-Ting Yi, and Martha Palmer. 2007. Combining lexical resources: mapping between propbank and verbnet. In *Proceedings of the 7th International Workshop on Computational Linguistics*, Tilburg, the Netherlands.

Mao, Chen, Dorer Klaus, Foroughi Ehsan, and Heintz Fredrik, 2003. *Users manual: RoboCup soccer server manual for soccer server version 7.07 and later*.

Màrquez, Lluís, Pere Comas, Jesús Giménez, and Neus Català. 2005. Semantic role labeling as sequential tagging. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 193–196, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Matsubayashi, Yuichiroh, Naoaki Okazaki, and Jun'ichi Tsujii. 2009. A comparative study on generalization of semantic roles in framenet. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 19–27, Suntec, Singapore, August. Association for Computational Linguistics.

McDonald, R., K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-2005)*, Ann Arbor, MI, USA, June.

Melli, G., Y. Wang, Y. Liu, M. Kashani, Z. Shi, B. Gu, A. Sarkar, and F. Popowich. 2005. Description of squash, the sfu question answering summary handler for the duc-2005 summarization task. In *Proceedings of Document Understanding Conferences (DUC-2005)*, pages 103–110, Vancouver, Canada, October.

Meyers, A., R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Crishman. 2004. Annotating noun argument structure for nombank. In *Proceedings of the fourth International Conference on Language Resources and Evaluation (LREC-2004)*.

Miller, George A. 1995. Wordnet: A lexical database for english. *Communications of the ACM*, 38:39–41.

Miller, S., H. Fox, L. Ramshaw, and R. Weischedel. 2000. A novel use of statistical parsing to extract information from text. *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2000)*, pages 226–233.

Miller, Scott, David Stallard, Robert Bobrow, and Richard Schwartz. 1996. A fully statistical approach to natural language interfaces. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL-1996)*, pages 55–61, Morristown, NJ, USA. Association for Computational Linguistics.

Moschitti, A. 2004. A study on convolution kernels for shallow semantic parsing. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-2004)*.

Moschitti, Alessandro, Daniele Pighin, and Roberto Basili. 2006. Semantic role labeling via tree kernel joint inference. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-2006)*.

Narayanan, S. and S. Harabagiu. 2004. Question answering based on semantic structures. In *Proceedings of 20th International Conference on Computational Linguistics (COLING-2004)*, University of Geneva, Switzerland, August 23rd-27th.

Palmer, M., D. Gildea, and P. Kingsbury. 2005. The proposition bank: an annotated corpus of semantic roles. *Computational Linguistics*, 31(1).

Palmer, M., J. Rosenzweig, and S. Cotton. 2001. Automatic predicate argument analysis of the penn treebank. In *Proceedings of the first International Conference on Human Language Technology Research*, San Diego.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pages 311–318, Morristown, NJ, USA. Association for Computational Linguistics.

Pazienza, Maria Teresa, Marco Pennacchiotti, and Fabio Massimo Zanzotto. 2006. Mixing wordnet, verbnet and propbank for studying verb relations. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC-2006)*.

Pollard, Carl and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press and CSLI Publications, Chicago, Illinois.

Ponzetto, S. P. and M. Strube. 2006. Semantic role labeling for coreference resolution. In *Proceedings of the 11st Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006)*, Trento, Italy, April 5–7.

Pradhan, S., K. Hacioglu, V. Krugler, W. Ward, J. H. Martin, and D. Jurafsky. 2005a. Support vector learning for semantic argument classification. *Machine Learning*.

Pradhan, S., K. Hacioglu, W. Ward, J. Martin, and D. Jurafsky. 2005b. Semantic role chunking combining complementary syntactic views. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL-2005)*, Ann Arbor, MI, USA, June.

Pradhan, S., K. Hacioglu, W. Ward, J. H. Martin, and D. Jurafsky. 2005c. Semantic role chunking combining complementary syntactic views. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, Ann Arbor, MI, USA, June.

Pradhan, S., W. Ward, K. Hacioglu, , J. H. Martin, and D. Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of Human Language Technology Conference / North American chapter of the Association for Computational Linguistics annual meeting (HLT-NAACL-2004)*.

Pradhan, S., W. Ward, K. Hacioglu, , J. H. Martin, and D. Jurafsky. 2005. Semantic role labeling using different syntactic views. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-2005)*, Ann Arbor, MI, USA, June.

Pradhan, Sameer, Wayne Ward, and James H. Martin. 2008. *Towards robust semantic role labeling*. Association for Computational Linguistics.

Punyakanok, V., D. Roth, and W. Yih. 2005a. Generalized inference with multiple semantic role labeling systems (shared task paper). In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL-2005)*, Ann Arbor, MI, USA.

Punyakanok, V., D. Roth, and W. Yih. 2005b. The necessity of syntactic parsing for semantic role labeling. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-2005)*, Edinburgh, Scotland, UK, July 30 – August 5.

Punyakanok, Vasin, Dan Roth, and Wen tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Rambow, Owen, K. Vijay-Shanker, and David Weir. 1995. D-tree grammars. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL-1995)*, pages 151–158, Morristown, NJ, USA. Association for Computational Linguistics.

Riloff, E. 1993. Automatically constructing a dictionary for information extraction tasks. In *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI-1993)*, Washington, DC, USA, July.

Riloff, E. and M. Schmelzenbach. 1998. An empirical approach to conceptual case frame acquisition. In *Proceedings of the Sixth Workshop on Very Large Corpora*.

Roth, D. and W. Yih. 2002. Probabilistic reasoning for entity relation & recognition. In *Proceedings of 19th International Conference on Computational Linguistics (COLING-2002)*, Taipei, Taiwan, August 24 - September 1.

Roth, Dan. 1998. Learning to resolve natural language ambiguities: a unified approach. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-1998)*, page 806, Madison, Wisconsin.

Schabes, Yves and Stuart M. Shieber. 1994. An alternative conception of tree-adjoining derivation. *Computational Linguistics*, 20(1):91–124.

Sekimizu, T., H.S. Park, and J. Tsujii. 1998. Identifying the interaction between genes and gene products based on frequently seen verbs in medline abstracts. In *Genome Informatics*, volume Examples of sentences. Shallow parser. Noun phrase recognizer. Algorithm to identify the arguments of verbs. 62-71.

Shen, L., L. Champollion, and A. Joshi. 2008. Ltag-spinal and the treebank: A new resource for incremental, dependency and semantic parsing. *Language Resources and Evaluation*, 42(1):1–19.

Shen, L. and A. Joshi. 2005. Building an LTAG TreeBank. Technical Report Technical Report MS-CIS-05-15,5, CIS Department, University of Pennsylvania.

Shen, L. and A. Joshi. 2008. Ltag dependency parsing with bidirectional incremental construction. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-2008)*.

Shen, L. and Aravind Joshi. 2005. Incremental ltag parsing. In *Proceedings of Human Language Technology Conference/Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005)*, Vancouver, British Columbia, Canada, October 6–8.

Shen, L., A. Sarkar, and A. Joshi. 2003. Using LTAG based features in parse reranking. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP-2003)*.

Shen, Libin. 2004. Nondeterministic ltag derivation tree extraction. In *Proceedings of the Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+7)*, Simon Fraser University, Vancouver, British Columbia, Canada, May 20-22.

Shen, Libin. 2006. *Statistical ltag parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA. Adviser-Joshi,, Aravind K.

Shi, Lei and Rada Mihalcea. 2005. Putting pieces together: combining framenet, verbnet and wordnet for robust semantic parsing. *Computational Linguistics and Intelligent Text Processing*, pages 100–111.

Shi, Z., A. Sarkar, and F. Popowich. 2007. Simultaneous identification of biomedical named-entity and functional relation using statistical parsing Techniques. In *Proceedings of Human Language Technologies/The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2007) (short paper)*.

Stone, Matthew and Christine Doran. 1997. Sentence planning as description using tree adjoining grammar. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics (ACL-EACL-2007)*, pages 198–205, Morristown, NJ, USA. Association for Computational Linguistics.

Sun, Weiwei, Zhifang Sui, and Meng Wang. 2009. Prediction of thematic rank for structured semantic role labeling. In *Proceedings of Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (short paper) (ACL-IJCNLP-2009)*, pages 253–256, Suntec, Singapore, August. Association for Computational Linguistics.

Surdeanu, M., S. Harabagiu, J. Williams, and P. Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-2003)*.

Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll 2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL-2008)*, pages 159–177, Manchester, England, August. Coling 2008 Organizing Committee.

Sutton, Charles and Andrew McCallum. 2005. Joint parsing and semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 225–228, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Suzuki, H. and K. Toutanova. 2006. Learning to predict case markers in japanese. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (COLING/ACL-2006)*.

Tateisi, Y., A. Yakushiji, T. Ohta, and J. Tsujii. 2005. Syntax annotation for the genia corpus. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing (IJCNLP-2005)*, Jeju Island, Korea, October 11-13.

Thompson, Cynthia A., Roger Levy, and Christopher D. Manning. 2003. A generative model for semantic role labeling. In *Proceedings of the 14th European Conference on Machine Learning (ECML-2003)*, Cavtat-Dubrovnik, Croatia, September 22-26.

Toutanova, K., A. Haghighi, and C. D. Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-2005)*, Ann Arbor, MI, USA, June.

Toutanova, K., A. Haghighi, and C.D. Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2).

Vickrey, David and Daphne Koller. 2008. Sentence simplification for semantic role labeling.

In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-2008:HLT)*, pages 344–352, Columbus, Ohio, June. Association for Computational Linguistics.

Wang, Yang and Greg Mori. 2009. Max-margin hidden conditional random fields for human action recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR-2009)*, Miami, FL, USA, June 20-26.

Weir, David Jeremy. 1988. *Characterizing mildly context-sensitive grammar formalisms.* Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA. Supervisor-Joshi, Aravind K.

Wong, Y.W. and R. J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of Human Language Technologies/The Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2006)*, New York, USA.

Wu, Dekai and Pascale Fung. 2009. Semantic roles for smt: a hybrid two-pass model. In *Proceedings of Human Language Technologies/The Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2009) (short paper)*, pages 13–16, Boulder, Colorado, June. Association for Computational Linguistics.

Xia, Fei. 1999. Extracting tree adjoining grammars from bracketed corpora. In *Proceedings of 5th Natural Language Processing Pacific Rim Symposium (NLPRS-1999)*, Beijing, China.

Xia, Fei. 2001. *Automatic grammar generation from two different perspectives.* Ph.D. thesis, University of Pennsylvania. Supervisor-Palmer, Martha and Supervisor-Joshi, Aravind.

Xue, N. and M. Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP-2004).*

Yakushiji, Akane, Yusuke Miyao, Yuka Tateisi, and Junichi Tsujii. 2005. Biomedical information extraction with predicate-argument structure patterns. In *Proceedings of the first International Symposium on Semantic Mining in Biomedicine (SMBM)*, Hinxton, Cambridgeshire, UK, April.

Yeh, A. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000)*, Universität des Saarlandes, Saarbrücken, Germany, July 31 - August 4.

Yi, S. and M. Palmer. 2005. The integration of syntactic parsing and semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Yi, Szu-Ting, Edward Loper, and Martha Palmer. 2007. Can semantic roles generalize across genres? In *Proceedings of the Human Language Technology Conferences/North American Chapter of the Association for Computational Linguistics Annual Meeting (HLT/NAACL-2007)*.

Yih, W. and K. Toutanova. 2006. Automatic semantic role labeling (tutorial). In *roceedings of Human Language Technology Conference/North American chapter of the Association for Computational Linguistics Annual Meeting (HLT-NAACL 2006)*, New York, USA.

Yu, Chun-Nam John and Thorsten Joachims. 2009. Learning structural svms with latent variables. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML-2009)*.

Zapirain, Beñat, Eneko Agirre, and Lluís Màrquez. 2008. Robustness and generalization of role sets: PropBank vs. VerbNet. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-2008:HLT)*, pages 550–558, Columbus, Ohio, June. Association for Computational Linguistics.

Zelenko, D., C. Aone, and A. Richardella. 2003. Kernel methods for relation extraction. *Journal of Machine Learning Research*.

Zettlemoyer, Luke and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL-2007)*, pages 678–687, Prague, Czech Republic, June. Association for Computational Linguistics.

Zettlemoyer, Luke S. and Michael Collins. 2005. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI-2005)*, pages 658–666, University of Edinburgh, Edinburgh, Scotland, July 26–29.

Zhang, Min, Jie Zhang, Jian Su, and Guodong Zhou. 2006. A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (COLING/ACL-2006)*, pages 825–832, Morristown, NJ, USA. Association for Computational Linguistics.