Master Thesis

Investigations into Semantic Role Labeling of PropBank and NomBank

by

Jiang Zheng Ping

Department of Computer Science

School of Computing

National University of Singapore

2006

Master Thesis

April 2006

INVESTIGATIONS INTO SEMANTIC ROLE LABELING OF PROPBANK AND NOMBANK

by

Jiang Zheng Ping

A THESIS SUBMITTED
FOR THE DEGREE OF MASTER OF SCIENCE
SCHOOL OF COMPUTING
NATIONAL UNIVERSITY OF SINGAPORE

Advisor: Dr. Ng Hwee Tou

ABSTRACT

The task of Semantic Role Labeling (SRL) concerns the determination of the generic semantic roles of constituents in a sentence. This thesis focuses on SRL based on the PropBank and NomBank corpora. Specifically, it addresses the following two questions:

- How do we exploit the interdependence of semantic arguments in a predicate-argument structure to improve an SRL system?
- How do we make use of the newly available NomBank corpus to build an SRL system that produces predicate-argument structures for nouns?

To address the first question, this thesis conducted experiments to explore various ways of exploiting the interdependence of semantic arguments to effectively improve the SRL accuracy on PropBank.

For the second question, this thesis adapted a PropBank-based SRL system to the SRL task of NomBank. Structures unique to NomBank's annotation are captured as additional features in a maximum entropy classification model to improve the adapted system.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Overview

The recent availability of semantically annotated corpora, such as FrameNet [Baker *et al.*, 1998][1], PropBank [Kingsbury *et al.*, 2002; Palmer *et al.*, 2005][2], NomBank [Meyers *et al.*, 2004d; 2004c][3] and various other semantically annotated corpora prompted research in automatically producing the semantic representations of English sentences. In this thesis, we study the semantic analysis of sentences based on PropBank and NomBank. For PropBank and NomBank, the semantic representation annotated is in the form of semantic roles, such as *ARG0*, *ARG1* for core arguments and *ARGM-LOC*, *ARGM-TMP* for modifying arguments of each predicate in a sentence. The annotation is done on the syntactic constituents in Penn TreeBank [Marcus *et al.*, 1993; Marcus, 1994] parse trees.

A sample PropBank and NomBank semantically labelled parse tree is presented in Figure 1.1. The PropBank predicate-argument structure labeling is underlined, while the labels of NomBank predicate-argument structure are given in italics. The PropBank verb predicate is "nominate", and its arguments are {(Ben Bernanke,

[1] http://framenet.icsi.berkeley.edu/

[2] http://www.cis.upenn.edu/~ace

[3] http://nlp.cs.nyu.edu/meyers/NomBank.html

ARG1), (Greenspan's replacement, ARG2), (last Friday, ARGM-TMP)}. The NomBank nominal predicate is "replacement", which has arguments {(Ben Bernanke, ARG0), (Greenspan's, ARG1), (last Friday, ARGM-TMP) }. It also has the special "support" verb "nominate", that introduces the argument (Bern Bernanke, ARG0).

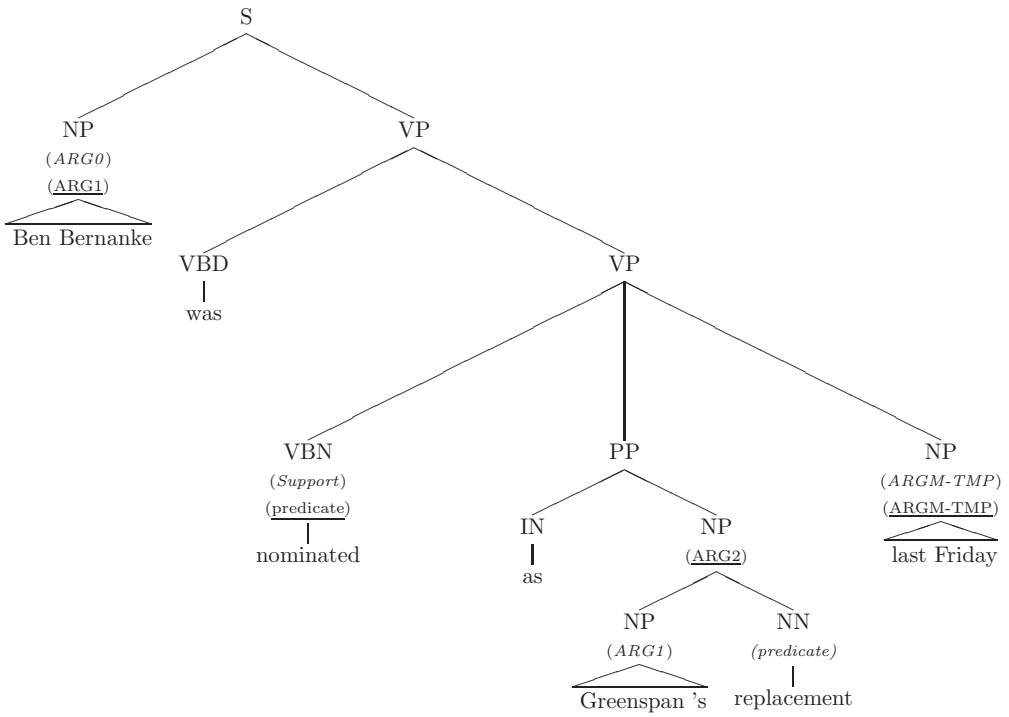

Figure 1.1: A sample syntactic parse tree labelled with PropBank and NomBank Semantic Arguments

The process of determining these semantic roles is known as Semantic Role Labeling (SRL). Most previous research uses machine learning techniques by treating the SRL task as a classification problem, and divides the task into two subtasks: *semantic argument identification* and *semantic argument classification*. Semantic argument identification involves classifying each syntactic element in a sentence into either a semantic argument or a non-argument. Semantic argument classification involves classifying each semantic argument identified into a specific semantic role.

This thesis documents initial explorations in improving PropBank based SRL accuracy and in building one of the first known NomBank based automatic SRL systems.

1.1 PropBank based SRL

Various features based on the syntactic structure of either shallow or full parse trees are used by previous work in building the classifier for semantic argument classification. These features capture the syntactic environment but overlook the semantic context of the argument currently being classified.

We propose a notion of *semantic context*, which consists of the already identified or role classified semantic arguments in the context of an argument that is being classified. Semantic context features are defined as features extracted from the neighboring arguments, and used in classifying the current argument. These features explicitly capture the interdependence among arguments of a predicate. An SVM-based classifier that exploits argument interdependence performs significantly better than a baseline classifier.

1.2 NomBank based SRL

We explore the possibility of adapting features previously shown useful in PropBank based SRL systems. Those features capture the structure of arguments and relationships between arguments and predicates. The adaptation is made by dropping certain features (such as the "voice" feature that denotes whether a verb predicate is active or passive voice) and changing the features regarding verb predicate to nominal predicate.

Various features specific to NomBank are proposed to augment the adapted fea-

ture set. These features try to capture the nominal predicates' classes as defined in NomBank, the nominal predicates' location with regard to support verbs, etc.

The large set of adapted and augmented features are subjected to a greedy feature selection algorithm based on each feature's performance contribution on a development data set. The experiments show the success of feature adaptation and the effectiveness of NomBank-specific features.

1.3 Contributions

This thesis aims to document the following two contributions.

- Empirically demonstrate the importance of capturing argument interdependence in analyzing verb predicate and argument structures. Propose an effective set of *semantic context* features that significantly improve PropBank based SRL system.
- Successfully adapt a PropBank SRL system to analyze noun predicate and argument structures. Provide one of the first known NomBank based SRL systems.

We hope this thesis serve as the basis for further investigation into the semantic understanding of natural languages, using PropBank and NomBank data with machine learning approaches.

1.4 Overview of this thesis

- Chapter 2 gives a brief review of recent research in SRL, with an emphasis on research based on PropBank and NomBank.

- Chapter 3 is a detailed explanation of work presented in [Jiang *et al.*, 2005], which exploits argument interdependence for PropBank SRL.
- Chapter 4 details one of the first attempts at building a NomBank based SRL system.
- Chapter 5 presents some possible future research directions.
- Chapter 6 concludes the thesis.

Chapter 2

Semantic Role Labeling: Previous Work

2.1 Construction of Semantically Annotated Tree-Banks

The recent research interests and activities in semantic analysis of natural language are partly fuelled by the construction of various semantically annotated corpora. The underlying motivation is to make possible systems that automatically produce semantic structures of sentences. These systems will play a key role in high precision Question Answering, Information Extraction, Machine Translation, and various other important Natural Language Processing applications.

The success of machine learning based syntactic parsers [Ratnaparkhi, 1998; Collins, 1999; 2000; Charniak, 2000; Charniak and Johnson, 2005] based on syntactically annotated treebanks is followed by research efforts at producing automatic semantic parsers. Here we review some major semantically annotated treebanks which form the basis of semantic parsers, or Semantic Role Labeling (SRL) systems. We emphasize

PropBank and NomBank, which are the basis of our developed SRL systems discussed in later chapters. Other notable semantic corpora include the FrameNet [Baker *et al.*, 1998][1] project at Berkeley and its various instances in other languages. [Ellsworth *et al.*, 2004] gives a comparison of PropBank, FrameNet, and FrameNet's German variant SALSA.

2.1.1 PropBank

PropBank [Kingsbury *et al.*, 2002; Palmer *et al.*, 2005; Palmer and Marcus, 2002][2] provides a semantic annotation layer on top of the Penn TreeBank [Marcus *et al.*, 1993; Marcus, 1994]. PropBank annotates the argument structures for verbal predicates only, and does not contain annotations for adjectives, deverbal nouns and predicate nominatives [Kingsbury *et al.*, 2002].

Core arguments of verbal predicates are numbered as *ARG0* to *ARG5*, depending on the valency of the predicate and the arguments' semantic roles. Generally, *ARG0* corresponds to agent and *ARG1* corresponds to direct recipient. The numbering of core arguments instead of assignment of specific thematic roles as done in FrameNet is designed to be theory-neutral and allows relatively simple mapping onto various linguistic theories [Kingsbury *et al.*, 2003]. Modifying arguments can be viewed as a more consistent re-annotation of the previous "functional tags" in Penn TreeBank. These arguments include *ARG-LOC* for location, *ARG-TMP* for time, etc.

2.1.2 NomBank

Similar to PropBank, NomBank [Meyers *et al.*, 2004d; 2004c][3] is also based on Penn TreeBank [Marcus *et al.*, 1993; Marcus, 1994]. NomBank extends PropBank by an-

[1] http://framenet.icsi.berkeley.edu/

[2] http://www.cis.upenn.edu/~ace

[3] http://nlp.cs.nyu.edu/meyers/NomBank.html

notating predicate-argument structures for nouns. Annotated nouns include verbal nominalizations, partitives, subject nominalizations, environmental nouns, relational nouns, and some other classes [Meyers *et al.*, 2004d]. Some special annotation constructions of NomBank are described below.

Support

Most of the noun predicate's arguments occur locally inside the noun phrase headed by the predicate. There are several cases when an argument is located outside the local noun phrase, including arguments introduced by support verbs, arguments across copulas, PP constructions, and etc [Meyers *et al.*, 2004a]. Here we give two examples of the "support verb" case, which constitute more than half of all the cases when an argument does not occur locally. The support verbs "made" and "planned" respectively introduce arguments "He" and "She" to the noun predicates "reservation" and "speech".

- He *made* a reservation of the hotel.
- She *planned* a speech at the University.

Hyphenated Modifiers

To annotate hyphenated phrases like "land-owner", NomBank treats "land" as *ARG1* and "owner" as both *ARG0* and *predicate*. The underlying unwrapped phrase is "The owner owns the land". NomBank uses HN to differentiate the many parts in a hyphenated phrase, where N is the numbering of the parts. In this example, "land" is annotated as *ARG1-H0*, "owner" as *ARG0-H1*.

2.2 Automatic Labeling Systems

We are not aware of any prior work on automatic SRL systems based on the recently available NomBank corpus. The work in [Pradhan *et al.*, 2004] presents an automatic SRL system for selected eventive nominalizations in FrameNet corpus. The SRL systems discussed below are all based on PropBank.

2.2.1 Different Machine Learning Methods

By treating the SRL task as a classification problem, many previous systems adopted standard machine learning based classifiers. Among the participating systems in the CoNLL 2004 and 2005 SRL competition [Carreras and Marquez, 2004; 2005], more than five different major learning algorithms were used. The learning algorithms include Maximum Entropy, Vector-based Linear Classifiers such as Support Vector Machines, Decision Trees, Memory-based Learning, and Transformation-based Error-driven Learning. Maximum Entropy and Vector-based Linear Classifiers are used by the majority of the existing SRL systems.

Many CoNLL 2005 [Carreras and Marquez, 2005] systems employed combinations of basic learning models through voting-like combination heuristics [Marquez *et al.*, 2005; Sang *et al.*, 2005], stacking of classifiers [Pradhan *et al.*, 2005a], Integer Linear Programming [Punyakanok *et al.*, 2005] and reranking [Haghighi *et al.*, 2005; Sutton and McCallum, 2005].

2.2.2 Different Features

Given only a sentence with a verb predicate, a human annotator might be able to identify and classify sequences of words as the predicate's arguments. But all of the previous effective SRL systems are based on feature sets that are much richer than

the mere word sequence in a sentence. The input data to a SRL system include words, POS tags, base chunks, clauses, named entities, and syntactic parse trees of the sentences. SRL systems then extract various features to be used during model training and testing. The feature sets used by most systems originate from and extend those used in [Gildea and Jurafsky, 2002]. The most common features include those in Table 2.1.

Feature	Meaning
predicate (Pr)	predicate lemma in the predicate-argument structures
voice (Vo)	grammatical voice of the predicate, either active or passive
subcat (Sc)	grammar rule that expands the predicate's parent node in the parse tree
phrase type (Pt)	syntactic category of the argument constituent
head word (Hw)	syntactic head of the argument constituent
path (Pa)	syntactic path through the parse tree from the argument constituent to the predicate constituent
position (Po)	relative position of the argument consitituent with respect to the predicate constituent, either left or right

Table 2.1: Basic features

Various SRL systems propose additional features which usually capture:

- More information inside or around the syntactic constituent, including the first and last word spanned by the constituent, the left and right sister of the constituent, and the parent of the constituent.
- Variation or function of the basic features, including partial path that captures only the ascending part of the "path" feature in Table 2.1, and tree distance that measures the length of the "path" feature. These features serve as back offs for the potentially sparse basic features.

- Specific features designed to improving the SRL of certain argument classes, including a feature that checks if the "head word" basic feature belongs to a set of words indicating time.

Maximum Entropy-based SRL systems presented in [Carreras and Marquez, 2004; 2005] mostly involve features that are combinations of the basic features in Table 2.1. [Xue and Palmer, 2004] reviews and analyzes the features used in SRL systems. Their empirical results demonstrate that feature engineering is important in effective SRL systems.

Chapter 3

PropBank based SRL

This chapter details work presented in [Jiang *et al.*, 2005]. The work is done in collaboration with Jia Li.

3.1 Introduction

Most previous research treats the semantic role labeling task as a classification problem, and divides the task into two subtasks: *semantic argument identification* and *semantic argument classification*. Semantic argument identification involves classifying each syntactic element in a sentence into either a semantic argument or a non-argument. A syntactic element can be a word in a sentence, a chunk in a shallow parse, or a constituent node in a full parse tree. Semantic argument classification involves classifying each semantic argument identified into a specific semantic role, such as $ARG0$, $ARG1$, etc.

Various features based on the syntactic structure of either shallow or full parse trees are used in building the classifier for semantic argument classification task. These features capture the syntactic environment but overlook the semantic context of the argument currently being classified.

We propose a notion of *semantic context*, which consists of the already identified or role classified semantic arguments in the context of an argument that is being classified. Semantic context features are defined as features extracted from the neighboring semantic arguments, and used in classifying the current semantic argument. These features explicitly capture the interdependence among the arguments of a predicate.

The semantic context features are applied in the full parse tree based PropBank semantic role labeling task. Experimental results show significant improvement in semantic argument classification accuracy over a purely local feature based classifier.

The rest of the Chapter is organized as follows: Section 3.2 explains in detail the application of semantic context features to semantic argument classification. Section 3.3 gives a specific example of how semantic context features can improve semantic argument classification and section 3.4 shows our experimental results. Section 3.5 reviews related work.

3.2 Semantic Context Based Argument Classification

Similar to [Pradhan *et al.*, 2005b], we treat the semantic role labeling task as a classification problem, and divide the task into semantic argument identification and semantic argument classification. In this section, we assume correct argument identification, and focus on argument classification. Section 3.4.8 will include results of argument classification integrated with identification.

3.2.1 Baseline Features

By treating the semantic role labeling task as a classification problem, one of the most important step in building an accurate classifier is feature selection. Most features

used in [Gildea and Jurafsky, 2002; Pradhan *et al.*, 2005b; Moldovan *et al.*, 2004; Bejan *et al.*, 2004; Thompson *et al.*, 2004] can be categorized into three types:

- *Sentence level features*, such as predicate, voice, and predicate subcategorization.
- *Argument-specific features*, such as phrase type, head word, content word, head word's part of speech, and named entity class of the argument.
- *Argument-predicate relational features*, such as an argument's position with respect to the predicate, and its syntactic path to the predicate.

The above features attempt to capture the syntactic environment of the semantic argument being identified or classified. They are entirely determined by the underlying full or shallow syntactic parse tree. They carry no information about those semantic arguments that have already been identified or classified. As such, the identification, as well as classification, of each semantic argument is done independently from one another. It assumes that the semantic arguments of the same predicate do not influence each other.

We use the baseline features in [Gildea and Jurafsky, 2002] and [Pradhan *et al.*, 2005b] as our baseline. These features are explained in Table 3.1.

3.2.2 Semantic Context Features

For a semantic argument, its semantic context features are defined as the features of its neighboring semantic arguments. The combination of the features from the argument itself and its neighboring arguments would encode interdependence among the arguments.

The semantically labeled example parse tree in Figure 3.1 annotates the predicate "added" and its arguments $ARG1$,$ARG2$, $ARG4$, and $ARGM\text{-}ADV$. Table 3.2 con-

Feature	Meaning
Sentence level features	
predicate (Pr)	predicate lemma in the predicate-argument structures
voice (Vo)	grammatical voice of the predicate, either active or passive
subcat (Sc)	grammar rule that expands the predicate's parent node in the parse tree
Argument specific features	
phrase type (Pt)	syntactic category of the argument constituent
head word (Hw)	syntactic head of the argument constituent
Argument-predicate relational features	
path (Pa)	syntactic path through the parse tree from the argument constituent to the predicate constituent
position (Po)	relative position of the argument consitituent with respect to the predicate constituent, either left or right

Table 3.1: Baseline features

tains baseline features (as defined in Table 3.1) extracted from each argument. When classifying argument $ARG2$, context features include path, phrase type, head word and position (respectively denoted as Pa, Pt, Hw and Po in Table 3.2) of arguments $ARG1$, $ARG4$ and $ARGM\text{-}ADV$. The previously classified semantic label (denoted as Ar), $ARG1$, is also part of the context features. Features Pr, Vo, and Sc, which are identical for each argument, are not part of the context features.

3.2.3 Various ways of Incorporating Semantic Context Features

There are combinatorially many subsets of the "context features" in Table 3.2 that one can choose to add to the baseline features. Argument ordering is also significant when each classification depends not only on its own features, but also on its neighboring

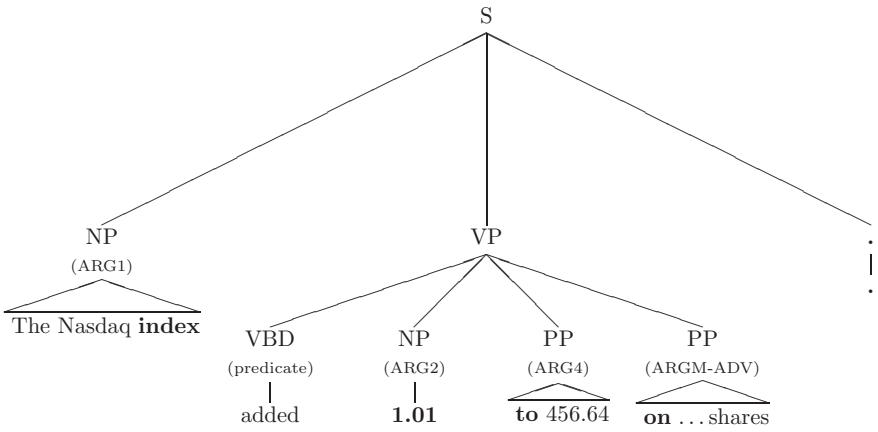

Figure 3.1: Semantically labeled parse tree, from dev set 00

non-context features			context features				
Pr	Vo	Sc	Pt	Hw	Pa	Po	Ar
add	active	VP:VBD_NP_PP_PP	NP	index	NP∧S_VP_VBD	L	ARG1
add	active	VP:VBD_NP_PP_PP	NP	1.01	NP∧VP_VBD	R	ARG2
add	active	VP:VBD_NP_PP_PP	PP	to	PP∧VP_VBD	R	ARG4
add	active	VP:VBD_NP_PP_PP	PP	on	PP∧VP_VBD	R	ARGM-ADV

Table 3.2: Semantic context features based on Figure 3.1

arguments. A good classifier should incorporate the most indicative classification ordering and set of context features.

We use context feature acronyms combined with subscripts to denote particular types of context features at specific locations with respect to the current argument being classified. e.g., Hw_1 refers to the head word of the immediate right neighboring argument. More examples are given in Table 3.3. We also use set notation $\{-i..i\}$ to denote the set of context features with subscript index $j \in \{-i..i\}$. e.g., $Hw_{\{-1..1\}}$ includes context feature Hw_{-1}, Hw_1.

feature	definition	example
Pt_i	the syntactic type of the ith context semantic element	Pt_{-1}=NP ; Pt_1=PP
Hw_i	the head word of the ith context semantic element	Hw_{-1}=index ; Hw_1=to
Pa_i	the path of the ith context semantic element	Pa_{-1}=NP$\wedge$S_VP_VBD ; Pa_1=PP$\wedge$VP_VBD
Po_i	the relative position of the ith context semantic element	Po_{-1}=L ; Po_1=R
Ar_i	the semantic label of the ith previous semantic element	Ar_{-1}=ARG1

Table 3.3: Semantic context features, capturing feature at the ith position with respect to the current argument. Examples are based on arguments in Figure 3.1, current argument is ARG2

Augmenting baseline with all types of Context Features

It is straightforward to incorporate all available context features. In Table 3.4, we assume a context feature set $\{-1..1\}$ with baseline features lightly shaded and additional context features darkly shaded.

Augmenting baseline with a single type of Context Features

We add only a specific type of context features to the baseline, in order to study in detail its effect. Table 3.5 shows adding features in $Hw_{\{-2..2\}}$ to the baseline.

Introducing a different Argument Ordering

So far we have implicitly assumed the *linear ordering* of classification, meaning the textual occurrence order by which each argument appears in a sentence. In PropBank, arguments of a single predicate in a sentence do not overlap, so this ordering is well-

non-context features			context features				
Pr	Vo	Sc	Pt	Hw	Pa	Po	Ar
add	active	VP:VBD_NP_PP_PP	NP	index	NP∧S_VP_VBD	L	ARG1
add	active	VP:VBD_NP_PP_PP	NP	1.01	NP∧VP_VBD	R	*
add	active	VP:VBD_NP_PP_PP	PP	to	PP∧VP_VBD	R	
add	active	VP:VBD_NP_PP_PP	PP	on	PP∧VP_VBD	R	

Table 3.4: Semantic context features based on Figure 3.1, adding all types of darkly shaded context features in set $\{-1..1\}$ to lightly shaded baseline features.

non-context features			context features				
Pr	Vo	Sc	Pt	Hw	Pa	Po	Ar
				nil			
add	active	VP:VBD_NP_PP_PP	NP	index	NP∧S_VP_VBD	L	ARG1
add	active	VP:VBD_NP_PP_PP	NP	1.01	NP∧VP_VBD	R	*
add	active	VP:VBD_NP_PP_PP	PP	to	PP∧VP_VBD	R	
add	active	VP:VBD_NP_PP_PP	PP	on	PP∧VP_VBD	R	
				nil			

Table 3.5: Semantic context features based on Figure 3.1, adding darkly shaded $Hw_{\{-2..2\}}$ to the lightly shaded baseline features.

defined. Linear ordering assumes language has semantic locality, such that arguments whose words are syntactically close are more correlated. Linear ordering of arguments in Figure 3.1 is $ARG1$, $ARG2$, $ARG4$, $ARGM\text{-}ADV$.

Inspired by [Lim *et al.*, 2004], we view a parse tree as containing subtrees of increasing size, each spanned by an ancestor node of the predicate tree node. *Subtree ordering* puts arguments under the smaller subtree before those spanned by the larger

subtree. Experiments in [Lim *et al.*, 2004] show that arguments spanned by the parent of predicate node can be more accurately classified. This could potentially benefit context features that look into the history of classification, e.g. previously classified argument classes. Subtree ordering of the arguments in Figure 3.1 is $ARG2$, $ARG4$, $ARGM\text{-}ADV$, $ARG1$.

3.3 Examples of the utility of Semantic Context Features

3.3.1 A detailed example

As suggested by [Kingsbury and Palmer, 2003], a crucial step in semantic role labeling is to determine the sense, or roleset as defined in PropBank, of the predicate verb. A PropBank predicate verb's roleset is largely based on its argument structure. During classification of a particular argument, the features of the surrounding arguments provide more evidence for the predicate's argument structure and its roleset than only the current argument's baseline features.

As an example, predicate "add" has three rolesets defined in PropBank, as shown in Table 3.6. The roleset for "add" in the example tree of Figure 3.1 is "add.03". During classification of the arguments in the example tree in Figure 3.1, the baseline classifier wrongly classifies $ARG2$ as $ARG1$ and $ARG4$ as $ARG2$. The misclassification might be caused by the baseline features resembling features of training samples belonging to roleset "add.02".

Experiments in section 3.4 show that a classifier based on baseline features augmented with $Hw_{\{-2..2\}}$ can correct the baseline classifier's misclassifications. Table 3.7 shows the occurrence frequency of each roleset of predicate "add". $count_1$ is total occurrence count, $count_2$ is constrained with "index" being the head word for the

Roleset	name	Arguments
add.01	say	ARG0(speaker), ARG1(utterance)
add.02	mathematics	ARG0(adder), ARG1(thing being added), ARG2(thing being added to)
add.03	achieve	ARG1(logical subject, patient, thing rising / gaining / being added to), ARG2(amount risen), ARG4(end point), ARGM-LOC(medium)

Table 3.6: The rolesets of predicate verb "add", defined in PropBank

first argument of predicate "add". The only 3 occurrences left in $count_2$ for roleset "add.03" shows how the head words of the surrounding arguments can be indicative of the predicate's roleset and thus its argument structure.

Linguistically, the improvement is achieved because the semantic context argument "The Nasdaq index" can not fill the role of "speaker" in roleset "add.01" or "adder" in roleset "add.02". And the baseline classifier's misclassification is due to arguments "1.01" and "to 456.64" filling the roles "thing being added" and "thing being added to" in roleset "add.02".

Besides head word, one can also study how other types of semantic context features constrain the predicate's roleset.

roleset	$count_1$	$count_2$
add.01	339	0
add.02	198	0
add.03	48	3

Table 3.7: Occurrence counts of role sets of "add" in PropBank data section 02-21.

3.3.2 Two more examples

The baseline classifier wrongly labels ARG2 in Figure 3.2 as ARG1. A closer study at the feature vectors of the arguments in Table 3.8 shows that ARG1 and ARG2 have almost identical baseline feature vector. With augmented semantic context features, the classifier can avoid labeling two ARG1 for one predicate.

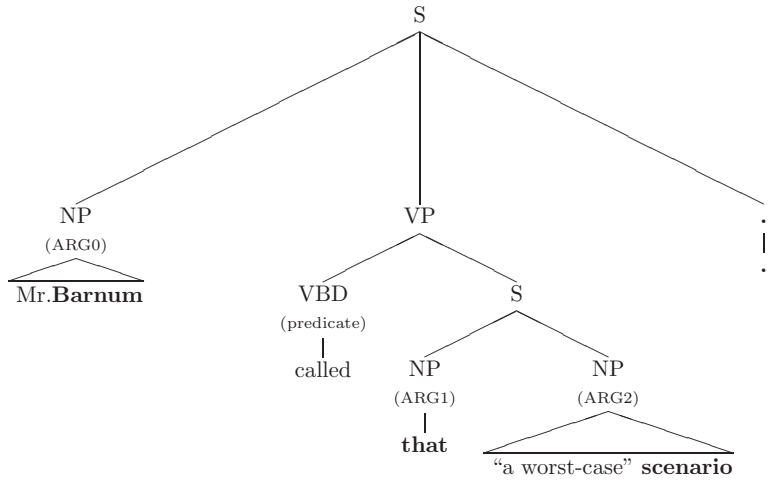

Figure 3.2: Semantically labeled parse tree, from dev set 00, 10th sentence in file wsj0018.mrg

non-context features			context features				
Pr	Vo	Sc	Pt	Hw	Pa	Po	Ar
call	active	VP:VBD_S	NP	barnum	NP∧S_VP_VBD	L	ARG0
call	active	VP:VBD_S	NP	that	NP∧S∧VP_VBD	R	ARG1
call	active	VP:VBD_S	NP	scenario	NP∧S∧VP_VBD	R	ARG2

Table 3.8: Semantic context features based on Figure 3.2

The semantic parse tree in Figure 3.3 and its feature vectors in Table 3.9 give an-

other example. Baseline classifier wrongly labels ARG2 and ARGM-TMP as ARGM-LOC and ARGM-LOC. Semantic context features can help avoid these confusions.

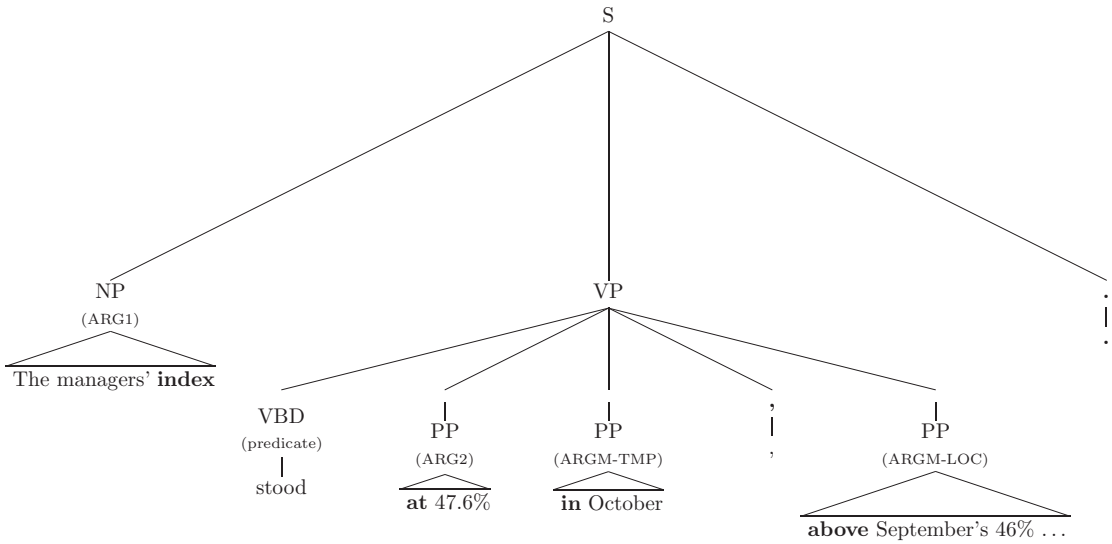

Figure 3.3: Semantically labeled parse tree, from dev set 00, 18th sentence in file wsj0059.mrg

non-context features			context features				
Pr	Vo	Sc	Pt	Hw	Pa	Po	Ar
stand	active	VP:VBD_PP_PP_,_PP	NP	index	NP∧S_VP_VBD	L	ARG1
stand	active	VP:VBD_PP_PP_,_PP	PP	at	PP∧VP_VBD	R	ARG2
stand	active	VP:VBD_PP_PP_,_PP	PP	in	PP∧VP_VBD	R	ARGM-TMP
stand	active	VP:VBD_PP_PP_,_PP	PP	above	PP∧VP_VBD	R	ARGM-LOC

Table 3.9: Semantic context features based on Figure 3.3

3.4 Experimental Results

The semantic argument classification task is done by yamcha[1] and tinysvm[2] [Kudo and Matsumoto, 2001]. Polynomial kernel with degree 2 is used in a one-vs-all classifier. The cost per unit violation of the margin $C = 1$, and tolerance of the termination criterion $e = 0.001$.

The training, development, and test data sections follow the conventional split of Section 02-21, 00, and 23 of PropBank release I respectively. In this section, we present accuracy scores of development section 00, unless otherwise noted. The argument classification accuracy using baseline features is 88.16%. Accuracy scores that have statistically significant improvement (χ^2 test with $p = 0.05$) over the baseline score are marked by "*". The best score of each row in the score tables is boldfaced, ties in score are broken arbitrarily.

3.4.1 Results based on Random Argument Ordering

To illustrate how argument classification ordering becomes important after semantic context features are introduced, as discussed in Section 3.2.3, we randomly permuted the argument order both in training and testing. Table 3.10 and Table 3.11 contain classification scores based on all and each type of semantic context features. None of the classification score based on randomly permuted argument order has statistically significant improvement over the baseline score of 88.16%.

A consistent argument ordering helps to establish a consistent semantic context for each argument during classification. We will see that classification based on a consistent ordering as in Section 3.4.2 and Section 3.4.3 perform significantly better than the random ordering here. Most of the best scores in each row of Table 3.12

[1] http://chasen.org/~taku/software/yamcha/

[2] http://chasen.org/~taku/software/TinySVM/

and 3.14 are significantly higher than that in the corresponding score in Table 3.10. Similar improvement is evident in Table 3.13 and 3.15 compared with Table 3.11.

	Accuracy of increasing context window size				
feature	{−1..1}	{−2..2}	{−3..3}	{−4..4}	{−5..5}
all	**87.61**	87.46	87.03	86.58	86.25
feature	{−1..0}	{−2..0}	{−3..0}	{−4..0}	{−5..0}
all	**87.07**	86.58	86.27	86.15	86.12
feature	{0..1}	{0..2}	{0..3}	{0..4}	{0..5}
all	**87.86**	87.34	87.11	86.88	86.66

Table 3.10: Accuracy based on adding all types of semantic context features, with increasing window size, assuming correct argument identification and random ordering of processing arguments

	Accuracy of increasing context window size				
feature	{−1..1}	{−2..2}	{−3..3}	{−4..4}	{−5..5}
Pt	**87.79**	87.53	87.28	87.10	86.78
Hw	88.36	**88.59**	88.30	87.98	87.82
Pa	**87.77**	87.69	87.39	87.15	86.96
Po	**87.58**	87.07	86.86	86.63	86.47
feature	{−1..0}	{−2..0}	{−3..0}	{−4..0}	{−5..0}
Ar	**87.52**	87.38	87.28	87.36	87.36

Table 3.11: Accuracy based on adding a single type of semantic context features, with increasing window size, assuming correct argument identification and random ordering of processing arguments

3.4.2 Results of Linear Ordering

Results using all types of context features

Table 3.12 contains experimental scores after augmenting baseline features with all types of semantic context features. *Ar* feature is only present within context feature sets that contain neighboring arguments to the left of the current argument. We notice from the accuracy scores that the context features of semantic arguments both to the left and right of the current argument are helpful when used together. Features of the arguments to the left of the current argument are not effective when used alone.

	Accuracy of increasing context window size				
feature	{−1..1}	{−2..2}	{−3..3}	{−4..4}	{−5..5}
all	89.27*	**89.32***	88.91	88.52	88.20
feature	{−1..0}	{−2..0}	{−3..0}	{−4..0}	{−5..0}
all	**87.88**	87.41	87.14	86.84	86.70
feature	{0..1}	{0..2}	{0..3}	{0..4}	{0..5}
all	88.74	**88.82**	88.48	88.32	88.08

Table 3.12: Accuracy based on adding all types of semantic context features, with increasing window size, assuming correct argument identification and linear ordering of processing arguments

Results using a single type of context features

Results of argument classification using baseline features augmented with a single type of context feature are shown in Table 3.13. The most salient semantic context feature is head word Hw. We attribute the negative effect of feature Ar to its susceptibility to previous argument classification errors, i.e., errors committed in position j in linear ordering might affect classification at position $> j$. However, a better argument

	Accuracy of increasing context window size				
feature	$\{-1..1\}$	$\{-2..2\}$	$\{-3..3\}$	$\{-4..4\}$	$\{-5..5\}$
Pt	89.01*	**89.06***	88.89	88.81	88.55
Hw	89.49*	**89.82***	89.56*	89.42*	89.32*
Pa	**89.17***	89.04*	88.89	88.66	88.55
Po	88.63	**88.74**	88.58	88.54	88.24
feature	$\{-1..0\}$	$\{-2..0\}$	$\{-3..0\}$	$\{-4..0\}$	$\{-5..0\}$
Ar	**87.98**	87.72	87.73	87.78	87.75

Table 3.13: Accuracy based on adding a single type of semantic context features, with increasing window size, assuming correct argument identification and linear ordering of processing arguments

classification ordering may improve the effect of *Ar*.

3.4.3 Results of Subtree Ordering

We repeat the experiments of the last subsection, but this time with the use of subtree ordering. The accuracy scores are given in Table 3.14 and Table 3.15. The most prominent difference from the results of linear ordering is the better accuracy scores with the use of the *Ar* features, as shown in Table 3.15 compared with Table 3.13. The improvement observed is consistent with the findings of [Lim *et al.*, 2004], that the argument class of the semantic arguments spanned by the parent of the predicate node can be more accurately determined.

3.4.4 More Experiments with *Ar* Feature

Unlike other semantic context features that are completely determined once argument identification is complete, the *Ar* feature is dynamically determined during argument

	Accuracy of increasing context window size				
feature	{−1..1}	{−2..2}	{−3..3}	{−4..4}	{−5..5}
all	**88.99***	88.96*	88.62	88.43	88.33
feature	{−1..0}	{−2..0}	{−3..0}	{−4..0}	{−5..0}
all	**88.92**	88.83	88.87	88.66	88.40
feature	{0..1}	{0..2}	{0..3}	{0..4}	{0..5}
all	**88.41**	88.38	88.08	87.80	87.63

Table 3.14: Accuracy based on adding all types of semantic context features, with increasing window size, assuming correct argument identification and subtree ordering of processing arguments

	Accuracy of increasing context window size				
feature	{−1..1}	{−2..2}	{−3..3}	{−4..4}	{−5..5}
Pt	**89.03***	88.98*	88.83	88.77	88.46
Hw	89.29*	**89.60***	89.29*	89.20*	89.08*
Pa	89.25*	**89.25***	89.14*	88.95	88.74
Po	88.77	**88.96***	88.72	88.37	88.21
feature	{−1..0}	{−2..0}	{−3..0}	{−4..0}	{−5..0}
Ar	88.85	88.84	**88.90**	88.87	88.83

Table 3.15: Accuracy based on adding a single type of semantic context features, with increasing window size, assuming correct argument identification and subtree ordering of processing arguments

classification and thus offers opportunity for a Beam Search algorithm to determine the globally optimal argument label sequence.

Ar Feature with the Gold Semantic Label

To explore the full potential of previous semantic classes Ar_i as semantic context features, we use the gold semantic label as previous classifications. The experimental results are in Table 3.16, titled "linear gold" and "subtree gold" respectively for linear and subtree ordering.

Using *Ar* with Beam Search

Tinysvm's output figures are converted to confidence scores between 0 and 1, by applying the Sigmoid function $y = \frac{1}{1+e^{-x}}$. We keep the top 3 most likely labels for each argument classification. Beam Search algorithm is then applied. The algorithm keeps multiple possible label sequences before the classification of all arguments is complete. It then finds and keeps the best k ($k = 10$ in our implementation) argument class sequences overall. Each label sequence is assigned a confidence score that is the product of scores of all arguments in the sequence. The sequence with the highest score is picked. Detailed algorithm is explained in Algorithm 1. Experimental results show improvements after using Beam Search with *Ar* feature, under "linear beam" and "subtree beam" in Table 3.16.

3.4.5 Combining Multiple Semantic Context Features

The best accuracy score we have obtained so far is 89.82%, given by $Hw_{-2..2}$ in Table 3.13. However, we want to leverage on the other types of semantic context features. Error analysis of experiments in Table 3.13 and 3.15 on development section 00 shows that classifiers using different semantic context features make different classification mistakes. This provides a basis for combining multiple semantic context features. Here we more carefully select the context features than simply using all available features as in Table 3.12 and 3.14.

Algorithm 1: Beam search for argument classification

Input $arg_{1..k}$ {a list of k un-classified arguments}

Initialize:
$M = 10$ {maximum number of completely classified argument list}
$N = 10$ {maximum number of partially classified argument list at each advance step}
$Q = 3$ {number of classes to keep in the beam for each arg_i}
$C = emptyheap$ {heap of completely classified argument list, sorted according to confidence score}
$S = emptyheap$ {heap of argument lists waiting to be advanced}
$T = emptyheap$ {heap of argument lists already advanced}
$insert(S, arg_{1..k})$

Beam search:
while $|C| < M$ and S not empty **do**
  $T = emptyheap$
  while $|T| < N$ **do**
    for each argument list $arg_{1..k}$ in $advance(extract(S), Q)$ **do**
      if $complete(arg_{1..k})$ **then**
        $insert(C, arg_{1..k})$
      else
        $insert(T, arg_{1..k})$
      end if
    end for
  end while
  S = T
end while

Return: $extract(C)$

Subroutine:
$advance(arg_{1..k}, Q)$: Classifies the next un-classified argument arg_i in $arg_{1..k}$, returns Q argument lists each with arg_i classified to one of Q most probable classes. Top Q most probable classes are chosen by confidence scores, from applying Sigmoid $y = \frac{1}{1+e^{-x}}$ to SVM classifier outputs. Confidence scores of arg_i's top Q classes are accumulated as a sumed log likelihood in the respective top Q $arg_{1..k}$ lists.
$insert(C, arg_{1..k})$: inserts $arg_{1..k}$ into heap C
$extract(C)$: extracts the most probable $arg_{1..k}$ from heap C
$complete(arg_{1..k})$: returns true if all arg_i where $1 \leq i \leq k$ have been classified

	Accuracy of increasing context window size				
Ar with	$\{-1..0\}$	$\{-2..0\}$	$\{-3..0\}$	$\{-4..0\}$	$\{-5..0\}$
linear	**87.98**	87.72	87.73	87.78	87.75
linear beam	**88.83**	88.46	88.64	88.66	88.67
linear gold	89.37*	89.54*	89.59*	**89.60***	89.58*
subtree	88.85	88.84	**88.90**	88.87	88.83
subtree beam	89.13*	**89.20***	89.18*	89.19*	89.20*
subtree gold	89.87*	89.84*	90.09*	90.07*	**90.11***

Table 3.16: More experiments with the *Ar* feature, using beam search and gold semantic label history

A Single Classifier with Multiple Context Features

The best context features from each row of Table 3.13, $Pt_{\{-2..2\}}$, $Hw_{\{-2..2\}}$, $Pa_{\{-1..1\}}$, $Po_{\{-2..2\}}$, and $Ar_{\{-1..1\}}$ are combined to form a classifier based on linear ordering. Similarly $Pt_{\{-1..1\}}$, $Hw_{\{-2..2\}}$, $Pa_{\{-2..2\}}$, $Po_{\{-2..2\}}$, and $Ar\{-3..3\}$ from Table 3.15 form a new subtree ordering based classifier. Experiments on development section 00 give accuracies of 89.54% and 89.19%, for linear and subtree ordering, respectively. The lack of accuracy improvement shows that a naive combination might accumulate the errors introduced by each additional context feature.

Voting with Multiple Classifiers

Instead of building a new classifier with multiple semantic context features, one can combine the classification results of multiple classifiers and hope to achieve better accuracy through consensus and mutual error correction. The voting process combines k different classifiers each belonging to one row in Table 3.13 or Table 3.15. Currently only classifiers based on the same ordering are combined. Classifiers are chosen from different context feature types, but can be of the same context feature window size.

For a particular semantic argument A, each candidate argument class will accumulate its confidence score assigned by all the voting classifiers. The candidate argument class with the highest aggregate score will be output as the argument class for A. Confidence score is obtained through the sigmoid function with the same approach as in Section 3.4.4. Details are presented in Algorithm 2. Exhaustive experiments are carried out with $k = 2, 3, 4, 5$.

On the development section, the best voting classifier group in linear ordering is $\{Pt_{\{-1..1\}},\ Hw_{\{-4..4\}},\ Pa_{\{-2..2\}},\ Ar_{\{-1..0\}}\}$, with accuracy 90.32%. The best for subtree ordering is $\{Pa_{\{-1..1\}}, Hw_{\{-2..2\}}, Ar_{\{-4..0\}}\}$with accuracy 90.62%. We denote the two voting classifiers as $Vote_{linear}$ and $Vote_{subtree}$ respectively.

Accuracy 90.62% of $Vote_{subtree}$ has statistically significant improvement over the best score we have obtained without voting, 89.82% of $Hw_{\{-2..2\}}$ in Table 3.13.

3.4.6 Accuracy on the Individual Argument Classes

We take the best voting classifier $Vote_{subtree}$ in Section 3.4.5 and analyze its performance on the top 20 most frequent argument classes in development section 00 of PropBank. The performance of each $Vote_{subtree}$'s component classifier is also recorded. The results are in Table 3.17.

Out of the top 10 most frequent argument classes, $Vote_{subtree}$ performs worse than the best of its component classifier only on *ARGM-LOC*. 7 of the 11th to 20th most frequent argument classes do not occur to benefit from voting. More intelligent voting based on weighting each classifier's confidence score with a prior could possibly improve the performance. We also observe that the classifiers are weaker for some specific argument classes. For argument class *ARGM-LOC*, although semantic context features help to improve the classification accuracy over the baseline, the overall accuracy still shows room for improvement. We leave these possibilities

Algorithm 2: Voting for argument classification

Input arg{argument being classified}
Input k{number of classifiers participating in voting, $k \leq 5$}
Input C{set of argument classes, $\{ARG0, ARG1...\}$}

Initialize:
for each c in C **do**
  $score[c] = 0$
end for
Choose k classifiers,
$\{M_1..M_k\} \subseteq \{Pt, Hw, Pa, Po, Ar\}$
for each M_i in $\{M_1..M_k\}$ **do**
  set window size w_i for M_i, $w_i \in \{1, 2, 3, 4, 5\}$
end for

Voting:
for each c in C **do**
  $score[c] = \sum_{i=1}^{k} ClassifierScore(M_i^{w_i}, arg, c)$
end for

Return: class c that maximizes $score[c]$

Subroutine:
$ClassifierScore(M_i^{w_i}, arg, c)$ applies classifier M_i with context windows size w_i to argument arg. It returns the confidence score of arg being classified as class c. Confidence score is obtained by applying Sigmoid function $y = \frac{1}{1+e^{-x}}$ to SVM classifier outputs.

to future research.

class	count	baseline	$Hw_{\{-2..2\}}$	$Pa_{\{-1..1\}}$	$Ar_{\{-4..0\}}$	$Vote_{subtree}$
ARG1	3459	94.13	95.26	94.94	95.00	**95.69**
ARG0	2391	96.45	97.28	97.03	96.36	**97.49**
R-ARG0	1014	95.27	97.63	96.35	96.75	**98.03**
R-ARG1	951	88.33	90.22	90.54	90.12	**93.17**
ARG2	920	85.43	85.76	87.07	86.96	**88.26**
ARGM-TMP	842	78.15	79.33	78.98	77.32	**80.88**
ARGM-MOD	383	99.74	99.74	**100.00**	99.74	**100.00**
ARGM-ADV	365	76.71	77.81	75.89	76.16	**78.63**
ARGM-LOC	347	58.79	**66.86**	60.52	59.65	65.13
C-ARG1	298	73.15	79.53	77.52	76.17	**81.88**
ARGM-MNR	285	61.40	59.30	62.46	60.35	**64.21**
ARGM-DIS	223	80.72	79.37	**83.86**	82.06	83.41
ARGM-NEG	180	**98.89**	97.22	**98.89**	**98.89**	98.33
ARG3	144	76.39	79.86	80.56	77.08	**81.25**
ARG4	108	78.70	77.78	77.78	**82.41**	79.63
ARGM-PNC	94	**70.21**	68.09	69.15	65.96	68.09
ARGM-CAU	68	61.76	**69.12**	64.71	58.82	66.18
R-ARG2	67	**58.21**	**58.21**	52.24	50.75	55.22
R-ARGM-TMP	56	83.93	**91.07**	85.71	89.29	**91.07**
ARGM-DIR	44	45.45	50.00	43.18	**54.55**	52.27

Table 3.17: Accuracy score for the top 20 most frequent argument classes in development section 00 of baseline classifier, $Vote_{subtree}$, and its component classifiers.

Linear ordering		Subtree ordering	
Feature	Accuracy	Feature	Accuracy
$Pt_{\{-2..2\}}$	89.26*	$Pt_{\{-1..1\}}$	88.88
$Hw_{\{-2..2\}}$	89.92*	$Hw_{\{-2..2\}}$	89.66*
$Pa_{\{-1..1\}}$	89.20*	$Pa_{\{-2..2\}}$	89.34*
$Po_{\{-2..2\}}$	88.96	$Po_{\{-2..2\}}$	89.20*
$Ar_{\{-1..0\}}$	88.14	$Ar_{\{-3..0\}}$	88.88
$Ar^{beam}_{\{-1..0\}}$	88.80	$Ar^{beam}_{\{-2..0\}}$	89.13*
$All_{\{-2..2\}}$	89.61*	$All_{\{-1..1\}}$	89.33*
$Vote_{linear}$	90.15*	$Vote_{subtree}$	90.50*

Table 3.18: Semantic argument classification accuracy on test section 23. Baseline accuracy is 88.41%

3.4.7 Testing on Section 23

Based on the experiments on development section 00, we choose the best classifiers and apply them to argument classification on test section 23. The baseline classification accuracy for Section 23 is 88.41%. Subtree voting improves the accuracy to 90.50%, representing a relative error reduction of 18%. The detailed scores are in Table 3.18.

3.4.8 Integrating Argument Classification with Baseline Identification

We implemented an argument identification system based on the baseline features used in [Pradhan *et al.*, 2005b]. Using 25% of the training data from Section 02-21, and adopt a "hard-prune" approach as defined in [Pradhan *et al.*, 2005b] during combined argument identification and classification. The F1 score of identification performed on test section 23 is 93.78%.

Feature	Precision	Recall	F1
baseline	84.44	83.01	83.72
$Vote_{linear}$	85.54*	84.10*	84.81*
$Vote_{subtree}$	85.95*	84.50*	85.22*

Table 3.19: Argument identification and classification score, test section 23

Table 3.19 presents the combined argument identification and classification score on test section 23. The baseline accuracy F1 score of 83.72% is established with the identification and classification system both held as baseline. The two best classifiers in Section 3.4.7 are then used with the baseline identification system to produce an improved integrated semantic role labeling system.

3.5 Related Work

To overcome the inadequate assumption of semantic argument independence during classification, [Punyakanok *et al.*, 2004] used the formalism of Integer Programming to impose global constraints during classifiers' decisions in semantic argument role labeling. The global constraints can be viewed as an introduction of semantic context knowledge, e.g., no duplicate argument classes. However, these constraints are rule based and do not easily generalize to linguistic exceptions or new languages. Our proposed semantic context features can be viewed as inducing the constraints by statistical learning.

Methods proposed in [Hacioglu *et al.*, 2004; Kouchnir, 2004; Park *et al.*, 2004] used features including the neighboring syntactic chunks' basic features such as phrase type and head word. We explicitly include features of the predicate's other semantic arguments, that might be syntactically far away.

Experiments in [Pradhan *et al.*, 2005b; Fleischman *et al.*, 2003; Kwon *et al.*, 2004]

used previous semantic arguments' role labels as dynamic context features and showed improved results. This demonstrates the utility of a limited semantic context feature. The semantic context features proposed here capture more extensively the interdependence among arguments. While [Pradhan *et al.*, 2005b] did not explicitly explain their ordering of argument classification, we showed that the ordering is important when previous semantic arguments' role labels are used.

Similarly, [Lim *et al.*, 2004] syntactically divided a sentence into the *immediate clause* and the *upper clauses*. Semantic arguments within the upper clauses are labeled after those in the immediate clause. This incremental approach allows semantic labels of the immediate clause to be used as additional features when labeling arguments within the upper clauses. Compared to their two level incremental approach, our proposed subtree ordering is more fine grained with exploration into different context feature window size.

This chapter extended previous attempts at capturing interdependence among the semantic arguments of a predicate. By combining a basic set of features with the notion of semantic context, and performing extensive experiments, we showed that semantic role labeling accuracies can be significantly improved using semantic context features.

Chapter 4

NomBank based SRL

This chapter is based on work presented in [Jiang and Ng, 2006].

4.1 Introduction

Automatic Semantic Role Labeling (SRL) systems, made possible by the availability of PropBank [Kingsbury and Palmer, 2003; Palmer *et al.*, 2005], and encouraged by development efforts in [Litkowski, 2004; Carreras and Marquez, 2004; 2005], have been shown to accurately determine the argument structure of verb predicates.

A successful PropBank-based SRL system would correctly determine that "Ben Bernanke" is the subject (labeled as ARG0 in PropBank) of predicate "replace", and "Greenspan" is the object (labeled as ARG1):

- Ben Bernanke replaced Greenspan as Fed chair.
- Greenspan was replaced by Ben Bernanke as Fed chair.

The recent release of NomBank [Meyers *et al.*, 2004d; 2004c], a databank that annotates argument structure for instances of common nouns in the Penn Treebank II

corpus, made it possible to develop automatic SRL systems that analyze the argument structures of noun predicates.

Given the following two noun phrases and one sentence, a successful NomBank-based SRL system should label "Ben Bernanke('s)" as the subject (ARG0) and "Greenspan('s)" as the object (ARG1) of the noun predicate "replacement".

- Greenspan's replacement Ben Bernanke
- Ben Bernanke's replacement of Greenspan
- Ben Bernanke was nominated as Greenspan's replacement.

The ability to automatically analyze the argument structures of verb and noun predicates would greatly facilitate NLP tasks like question answering, information extraction, etc. PropBank based SRL system as presented in Chapter 3 can not analyze predicate-argument structures that do not contain verb predicates, but phrases like "Ben Bernanke's replancement of Greenspan " are abundant in free texts.

This chapter focuses on our efforts at building an accurate automatic NomBank-based SRL system. We study how techniques used in building PropBank SRL system can be transferred to developing NomBank SRL system. We also make NomBank-specific enhancements to our baseline system. Our implemented SRL system and experiments are based on the September 2005 release of NomBank (NomBank.0.8).

The rest of this chapter is organized as follows: Section 4.2 gives an overview of NomBank, Section 4.3 introduces the Maximum Entropy classification model, Section 4.4 introduces our features and feature selection strategy, Section 4.5 explains the experimental setup and presents the experimental results, Section 4.6 compares NomBank SRL to PropBank SRL, and discusses the difficulties in NomBank SRL.

4.2 Overview of NomBank

The NomBank [Meyers *et al.*, 2004d; 2004c] annotation project originated from the NOMLEX [Macleod *et al.*, 1997; 1998] nominalization lexicon developed under the NYU Proteus Project. NOMLEX lists 1,000 nominalizations and the correspondences between their arguments and the arguments of their verb counterparts. NomBank frames combine various lexical resources [Meyers *et al.*, 2004b], including an extended NOMLEX and PropBank frames, and form the basis for annotating the argument structures of common nouns.

Similar to PropBank, NomBank annotation is made on the Penn TreeBank II (PTB II) corpus. For each common noun in PTB II that takes arguments, its core arguments are labeled with ARG0, ARG1, etc, and modifying arguments are labeled with ARGM-LOC to denote location, ARGM-MNR to denote manner, etc. Annotations are made on PTB II parse tree nodes, and argument boundaries align with the span of parse tree nodes.

A sample sentence and its parse tree labeled in the style of NomBank is shown in Figure 4.1. For the nominal predicate "replacement", "Ben Bernanke" is labeled as ARG0 and "Greenspan's" is labeled as ARG1. There is also the special label "Support" on "nominated" which introduces "Ben Bernanke" as an argument of "replacement". The support construct will be explained in detail in Section 4.4.2.

4.3 Model training and testing

We treat the NomBank-based SRL task as a classification problem and divide it into two phases: argument identification and argument classification. During the argument identification phase, each parse tree node is marked as either argument or non-argument. Each node marked as argument is then labeled with a specific

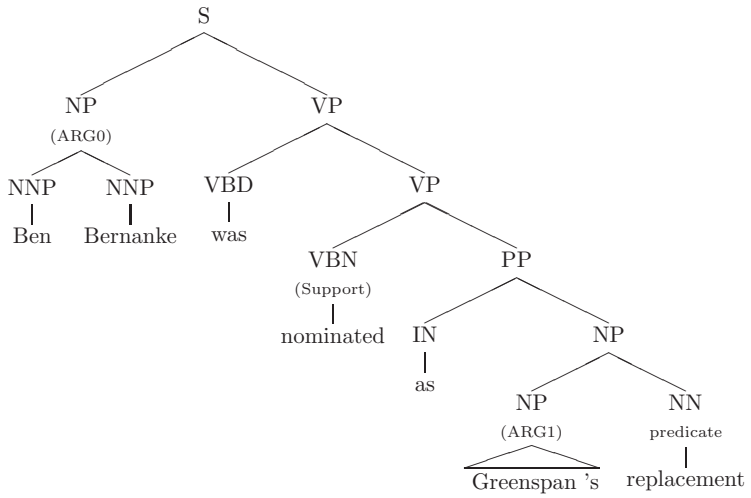

Figure 4.1: A sample sentence and its parse tree labeled in the style of NomBank

class during the argument classification phase. The identification model is a binary classifier , while the classification model is a multi-class classifier.

Opennlp maxent[1], an implementation of Maximum Entropy (ME) modeling, is used as the classification tool. Since its introduction to the Natural Language Processing (NLP) community [Berger *et al.*, 1996], ME-based classifiers have been shown to be effective in various NLP tasks. ME modeling is based on the insight that the best model is consistent with the set of constraints imposed and otherwise as uniform as possible. ME models the probability of label l given a vector of features x as in Equation 4.1. $f_i(l, x)$ is the binary feature function that maps label l and feature vector x to either 0 or 1, while the summation is over all n feature functions and with λ_i as the weight parameter for each feature function. Z_x is a normalization factor. In the identification model, label l corresponds to either "argument" or "non-argument", and in the classification model, label l corresponds to one of the specific NomBank argument classes. The classification result is the label l with the highest probability

[1]http://maxent.sourceforge.net/

$p(l|x)$.

$$p(l|x) = \frac{exp(\sum_{i=1}^{n} \lambda_i f_i(l, x))}{Z_x} \tag{4.1}$$

To train the ME-based identification model, training data is gathered by treating each parse tree node that is an argument as a positive example and the rest as negative examples. Classification training data is generated from argument nodes only.

During testing, the algorithm of enforcing non-overlapping arguments by [Toutanova *et al.*, 2005] is used. The algorithm maximizes the log-probability of the entire NomBank labeled parse tree. Specifically, assuming we only have two classes "ARG" and "NONE", the log-probability of a NomBank labeled parse tree is defined by Equation 4.2. $Max(T)$ is the maximum log-probability of a tree T, $NONE(T)$ and $ARG(T)$ are respectively the log-probability of assigning label "NONE" and "ARG" by our argument identification model to tree node T, *child* ranges through each of T's children, and $NONETree(child)$ is the log-probability of each node that is dominated by node *child* being labeled as "NONE". Details are presented in Algorithm 3.

$$Max(T) = max \begin{cases} NONE(T) + \sum(Max(child)) \\ ARG(T) + \sum(NONETree(child)) \end{cases} \tag{4.2}$$

NomBank sections 02-21 are used as training data, section 24 and 23 are respectively used as development and testing data.

4.3.1 Training data preprocessing

Unlike PropBank annotation which does not contain overlapping arguments (in the form of parse tree nodes domination) and does not allow predicates to be dominated by arguments, NomBank annotation in the September 2005 release contains such cases. In NomBank sections 02-21, about 0.6% of the argument nodes dominate

Algorithm 3: Maximize probability of Semantic Role labelled tree

Input p{syntactic parse tree}
Input m{argument identification model, assigns each constituent in the parse tree log likelihood of being a semantic argument}
Output score{maximum log likelihood of the parse tree with arguments identified}

MLparse(p, m)
if parse p is leaf node **then**
  return $max(Score(p, m, \text{“}ARG\text{”}), Score(p, m, \text{“}NONE\text{”}))$
else
  $childrenMLscore = 0$
  for each node c_i in $Children(p)$ **do**
    $childrenMLscore \mathrel{+}= MLparse(c_i, m)$
  end for
  $childrenNONEscore = 0$
  for each node c_i in $Children(p)$ **do**
    $childrenNONEscore \mathrel{+}= NONEARGparse(c_i, m)$
  end for
  return $max(Score(p, m, \text{“}NONE\text{”}) + childrenMLscore$,
  $Score(p, m, \text{“}ARG\text{”}) + childrenNONEscore)$
end if

NONEARGparse(p,m)
$allNONEscore = Score(p, m, \text{“}NONE\text{”})$
if parse p is leaf node **then**
  return $allNONEscore$
else
  for each node c_i in $Children(p)$ **do**
    $allNONEscore \mathrel{+}= NONEARGparse(c_i, m)$
  end for
  return $allNONEscore$
end if

Subroutine:
$Children(p)$ returns the list of children nodes of p.
$Score(p, m, state)$ returns the log likelihood assigned by model m, for parse p with *state*. *state* is either “ARG” or “NONE”.

some other argument nodes or the predicate. To simplify our task, during training example generation, we ignore arguments that dominate the predicate. We also

ignore arguments that are dominated by other arguments, so that when argument domination occurs, only the argument with the largest word span is kept. We do *not* perform similar pruning on the test data.

4.4 Features and feature selection

4.4.1 Baseline NomBank SRL features

Table 4.1 lists the baseline features we adapted from previous PropBank-based SRL systems [Pradhan *et al.*, 2005b; Xue and Palmer, 2004]. For ease of description, related features are grouped, with a specific individual feature given individual reference name. For example, feature b11FW in the group b11 denotes the first word spanned by the constituent and b13LH denotes the left sister's head word. We also experimented with various feature combinations, inspired by the features used in [Xue and Palmer, 2004]. These are listed as features b31 to b34 in Table 4.1.

Suppose the current constituent under identification or classification is "NP-Ben Bernanke" in Figure 4.1. The values of the baseline features in Table 4.1 are presented in Table 4.2. The symbol "NULL" is used to denote features that fail to instantiate.

4.4.2 NomBank-specific features

NomBank predicate morphology and class

The "NomBank-morph" dictionary provided by the current NomBank release maps the base form of a noun to various morphological forms. Besides singular-plural noun form mapping, it also maps base nouns to hyphenated and compound nouns. e.g., "healthcare" and "medical-care" both map to "care". For NomBank SRL features, we use this set of more specific mappings to replace the morphological mappings

Baseline Features [Pradhan *et al.*, 2005b]	
b1	predicate: stemmed noun
b2	subcat: grammar rule that expands the predicate's parent
b3	phrase type: syntactic category of the constituent
b4	head word: syntactic head of the constituent
b5	path: syntactic path from the constituent to the predicate
b6	position: to the left or right of the predicate
b11	first or last word/POS spanned by the constituent (b11FW, b11LW, b11FP, b11LP)
b12	phrase type of the left or right sister (b12L, b12R)
b13	left or right sister's head word/POS (b13LH, b13LP, b13RH, b13RP)
b14	phrase type of parent
b15	parent's head word or its POS (b15H, b15P)
b16	head word of the constituent if its parent has phrase type PP
b17	head word or POS tag of the rightmost NP node, if it is PP (b17H, b17P)
b18	phrase type appended with the length of path
b19	temporal keyword, e.g., "Monday"
b20	partial path from the constituent to the lowest common ancestor with the predicate
b21	projected path from the constituent to the highest NP dominating the predicate
Baseline Combined Features [Xue and Palmer, 2004]	
b31	b1 & b3
b32	b1 & b4
b33	b1 & b5
b34	b1 & b6

Table 4.1: Baseline features experimented in statistical NomBank SRL

based on WordNet. Specifically, we replace feature b1 in Table 4.1 with feature a1 in Table 4.3.

The current NomBank release also contains the "NOMLEX-PLUS" dictionary, which contains the class of nominal predicates according to their origin and the roles they play. e.g., "employment" originates from the verb "employ" and is classified as

Baseline Features [Pradhan *et al.*, 2005b]	
b1	replacement
b2	NP → NP NN
b3	NP
b4	Bernanke
b5	NP↑S↓VP↓VP↓PP↓NP↓NN
b6	left
b11	Ben, Bernanke, NNP, NNP
b12	NULL, VP
b13	NULL, NULL, was, VBD
b14	S
b15	was, VBD
b16	NULL
b17	NULL, NULL
b18	NP-7
b19	NULL
b20	NP↑S
b21	NP↑S↓VP↓VP↓PP↓NP
Baseline Combined Features [Xue and Palmer, 2004]	
b31	replacement & NP
b32	replacement & Bernanke
b33	replacement & NP↑S↓VP↓VP↓PP↓NP↓NN
b34	replacement & left

Table 4.2: Baseline feature values, assuming the current constituent is "NP-Ben Bernanke" in Figure 4.1.

"VERB-NOM", while the nouns "employer" and "employee" are classified as "SUBJECT" and "OBJECT" respectively. Other classes include "ADJ-NOM" for nominalization of adjectives and "NOM-REL" for relational nouns. The class of a nominal predicate is very indicative of the role of its arguments. We would expect a "VERB-NOM" predicate to take both ARG0 and ARG1, while an "OBJECT" predicate to

Additional Features Based on NomBank	
a1	Nombank morphed noun stem
a2	Nombank nominal class
a3	identical to predicate?
a4	a DEFREL noun?
a5	whether under the noun phrase headed by the predicate
a6	whether the noun phrase headed by the predicate is dominated by a VP node or has neighboring VP nodes
a7	whether there is a verb between the constituent and the predicate
Additional Combined Features	
a11	a1 & a2
a12	a1 & a3
a13	a1 & a5
a14	a3 & a4
a15	a1 & a6
a16	a1 & a7
Additional Features of Neighboring Arguments	
n1	for each argument already classified, b3-b4-b5-b6-r, where r is the argument class, otherwise b3-b4-b5-b6
n2	backoff version of n1, b3-b6-r or b3-b6

Table 4.3: Additional NomBank-specific features for statistical NomBank SRL

take only ARG0. We incorporated the class of nominal predicates as additional features in our NomBank SRL system. We add feature a2 in Table 4.3 to use this information.

DEFREL relational noun predicate

About 14% of the argument node instances in NomBank sections 02-21 are identical to their nominal predicate nodes. Most of these nominal predicates are DEFREL re-

lational nouns [Meyers *et al.*, 2004d]. Examples of DEFREL relational nouns include "employee", "participant", and "husband", where the nominal predicate itself takes part as an implied argument.

We include in our classification features an indicator of whether the argument coincides with the nominal predicate. We also include a feature testing if the argument is one of the DEFREL nouns we extracted from NomBank training sections 02-21. These two features correspond to a3 and a4 in Table 4.3.

Support verb

Statistics show that almost 60% of the arguments of nominal predicates occur locally inside the noun phrase headed by the nominal predicate. For the cases where an argument appears outside the local noun phrase, over half of these arguments are introduced by support verbs. Consider our example "Ben Bernanke was nominated as Greenspan's replacement.", the argument "Ben Bernanke" is introduced by the support verb "nominate". The arguments introduced by support verbs can appear syntactically distant from the nominal predicate.

To capture the location of arguments and the existence of support verbs, we add features indicating whether the argument is under the noun phrase headed by the predicate, whether the noun phrase headed by the predicate is dominated by a VP phrase or has neighboring VP phrases, and whether there is a verb between the argument and the predicate. These are represented as features a5, a6 and a7 in Table 4.3.

We also experimented with various feature combinations, inspired by the features used in [Xue and Palmer, 2004]. These are listed as features a11 to a16 in Table 4.3.

Neighboring arguments

Our work [Jiang *et al.*, 2005] and that of [Toutanova *et al.*, 2005] have shown the importance of capturing information of the global argument frame in order to correctly classify the local argument.

We make use of the the features {b3,b4,b5,b6} of the neighboring arguments as defined in Table 4.1. Arguments are classified from left to right in the textual order they appear. For arguments that are already labeled, we also add their argument class r. Specifically, for each argument to the left of the current argument, we have a feature b3-b4-b5-b6-r. For each argument to the right of the current argument, the feature is defined as b3-b4-b5-b6. We extract features in a window of size 7, centered at the current argument. We also add a backoff version (b3-b6-r or b3-b6) of this specific feature. These additional features are shown as n1 and n2 in Table 4.3.

Suppose the current constituent under identification or classification is "NP-Ben Bernanke". The values of the additional features in Table 4.3 are listed in Table 4.4.

4.4.3 Feature selection

Features used by our SRL system are automatically extracted from PTB II parse trees manually labeled in NomBank. Features from Table 4.1 and Table 4.3 are selected empirically and incrementally according to their contribution to test accuracy on the development section 24. The feature selection stops when adding any of the remaining features fail to improve the SRL accuracy on development section 24. We start the selection process with the basic set of features {b1,b2,b3,b4,b5,b6}. The detailed feature selection algorithm is presented in Algorithm 4.

Features for argument identification and argument classification are independently selected. To select the features for argument classification, we assume that all arguments have been correctly identified.

Additional Features based on NomBank	
a1	replacement
a2	VERB-NOM
a3	no
a4	no
a5	no
a6	yes
a7	yes
Additional Combined Features	
a11	replacement & VERB-NOM
a12	replacement & no
a13	replacement & no
a14	no & no
a15	replacement & yes
a16	replacement & yes
Additional Features of Neighboring Arguments	
n1	NP-Greenspan-NP↑NP↓NN-left
n2	NP-left

Table 4.4: Additional feature values, assuming the current constituent is "NP-Ben Bernanke" in Figure 4.1.

After performing greedy feature selection, the baseline set of features selected for identification is {b1-b6, b11FW, b11LW, b12L, b13RH, b13RP, b14, b15H, b18, b20, b32-b34}, and the baseline set of features selected for classification is {b1-b6, b11, b12, b13LH, b13LP, b13RP, b14, b15, b16, b17P, b20, b31-b34}. Note that features in {b19, b21} are not selected. For the additional features in Table 4.3, greedy feature selection chose {a1, a5, a6, a11, a12, a14} for the identification model and {a1, a3, a6, a11, a14, a16, n1, n2} for the classification model.

Algorithm 4: Greedy Feature Selection

Input $F_{candidate}$\{all candidate feature set\}
Output F_{select}\{selected feature set\}
Output M_{select}\{selected model\}

Initialize:
$F_{select} = \{b1, b2, b3, b4, b5, b6\}$
$F_{candidate} = AllFeatures - F_{select}$

$M_{select} = Train(F_{select})$
$E_{select} = Evaluate(M_{select}, DevData)$
loop
  for each feature f_i in $F_{candidate}$ **do**
    $F_i = F_{select} \bigcup f_i$
    $M_i = Train(F_i)$
    $E_i = Evaluate(M_i, DevData)$
  end for
  $E_{max} = Max(E_i)$
  if $E_{max} > E_{select}$ **then**
    $F_{select} = F_{select} \bigcup f_{max}$
    $M_{select} = M_{max}$
    $E_{select} = E_{max}$
    $F_{candidate} = F_{candidate} - f_{max}$
  end if
  if $F_{candidate} == \phi$ or $E_{max} \leq E_{select}$ **then**
    return F_{select}, M_{select}
  end if
end loop
Subroutine:
$Evaluate(Model, Data)$ returns the accuracy score by evaluating Model on Data.
$Train(FeatureSet)$ returns Maxent model trained on the given feature set.

4.5 Experimental result

4.5.1 Score on development section 24

After applying the feature selection algorithm in Section 4.4.3, the SRL F1 scores on the development section 24 are presented in Table 4.5. We separately present

the F1 score for identification-only and classification-only model. We also apply the classification model on the output of the identification phase (which may contain erroneouly identified arguments in general) to obtain the combined accuracy. During identification-only and combined identification and classification testing, the tree log-probability maximization algorithm based on Equation 2 (and its extension to multi-classes) is used. During the classification-only testing, we classify each correctly identified argument independently using the classification ME model. The "baseline" row lists the F1 scores when only the baseline features are used, and the "additional" row lists the F1 scores when additional features are added to the baseline features.

	identification	classification	combined
baseline	80.32	84.86	69.70
additional	80.55	87.31	70.12

Table 4.5: NomBank SRL F1 scores on development section 24

4.5.2 Testing on section 23

The identification and classification models based on the chosen features in Section 4.4.3 are then applied to test section 23. The resulting F1 scores are listed in Table 4.6. Using additional features, the identification-only, classification-only, and combined F1 scores are 82.50, 87.80, and 72.73, respectively. The detailed score for each argument class, from the combined identification and classification test is presented in Table 4.7.

Performing Chi-square test at the level of significance 0.05, we found that the improvement of the classification model using additional features compared to using just the baseline features is statistically significant, while the corresponding improvements due to additional features for the identification model and the combined model

are not statistically significant.

The improved classification accuracy due to the use of additional features does not contribute any significant improvement to the combined identification and classification SRL accuracy. This is due to the noisy arguments identified by the inadequate identification model, since the accurate determination of the additional features (such as those of neighboring arguments) depend critically on an accurate identification model.

	identification	classification	combined
baseline	82.33	85.85	72.20
additional	82.50	87.80	72.73

Table 4.6: NomBank SRL F1 scores on test section 23

4.5.3 Using automatic syntactic parse

So far we have assumed the availability of correct syntactic parse trees during model training and testing. We relax this assumption by using the re-ranking parser presented in [Charniak and Johnson, 2005] to automatically generate the syntactic parse trees for both training and test data.

The F1 scores of our best NomBank SRL system, when applied to automatic syntactic parse trees, are 66.77 for development section 24 and 69.14 for test section 23. These F1 scores are for combined identification and classification, with the use of additional features. Comparing these scores with those in Table 4.5 and Table 4.6, the usage of automatic parse trees lowers the F1 accuracy by more than 3%. The decrease in accuracy is expected, due to the noise introduced by automatic syntactic parsing.

4.6 Discussion

4.6.1 Comparison of the composition of PropBank and NomBank

Counting the number of annotated predicates, the size of NomBank.0.8 is about 83% of PropBank release 1. Preliminary consistency tests reported in [Meyers *et al.*, 2004d] shows that NomBank's inter-annotator agreement rate is about 85% for core arguments and lower for adjunct arguments. The inter-annotator agreement for PropBank reported in [Palmer *et al.*, 2005] is above 0.9 in terms of the Kappa statistic [Sidney and Castellan Jr., 1988]. While the two agreement measures are not directly comparable, the current NomBank.0.8 release documentation indicates that only 32 of the most frequently occurring nouns in PTB II have been adjudicated.

We believe the smaller size of NomBank.0.8 and the potential noise contained in the current release of the NomBank data may partly explain our lower SRL accuracy on NomBank, especially in the argument identification phase, as compared to the published accuracies of PropBank based SRL systems.

4.6.2 Difficulties in NomBank SRL

The argument structure of nominalization phrases is less fixed (i.e., more flexible) than the argument structure of verbs. Consider again the example given in the introduction, we find the following flexibility in forming grammatical NomBank argument structures for "replacement":

- The positions of the arguments are flexible, so that "Greenspan's replacement Ben Bernanke", "Ben Bernanke's replacement of Greenspan" are both grammatical.

- Arguments can be optional, so that "Greenspan's replacement will assume the post soon", "The replacement Ben Bernanke will assume the post soon", and "The replacement has been nominated" are all grammatical.

With the verb predicate "replace", except for "Greenspan was replaced", there is no freedom of forming phrases like "Ben Bernanke replaces" or simply "replaces" without supplying the necessary arguments to complete the grammatical structure.

We believe the flexible argument structure of NomBank noun predicates contributes to the lower automatic SRL accuracy as compared to that of the PropBank SRL task.

```
Number of Sentences    :        2416
Number of Propositions :        4502
Percentage of perfect props :  50.20

              corr.  excess  missed    prec.    rec.      F1
------------------------------------------------------------
   Overall     6334    2153    2597    74.63   70.92   72.73
----------
      ARG0     1502     370     564    80.24   72.70   76.28
      ARG1     2370     710     710    76.95   76.95   76.95
      ARG2      907     298     352    75.27   72.04   73.62
      ARG3      223      52      88    81.09   71.70   76.11
      ARG4        7       3      10    70.00   41.18   51.85
      ARG5        0       0       1     0.00    0.00    0.00
      ARG8        2       1       3    66.67   40.00   50.00
      ARG9        0       0       2     0.00    0.00    0.00
  ARGM-ADV        3       3      11    50.00   21.43   30.00
  ARGM-CAU        4       2       4    66.67   50.00   57.14
  ARGM-DIR        0       0       1     0.00    0.00    0.00
  ARGM-DIS        0       0       1     0.00    0.00    0.00
  ARGM-EXT       33      14      26    70.21   55.93   62.26
  ARGM-LOC      128      80      90    61.54   58.72   60.09
  ARGM-MNR      277     153     100    64.42   73.47   68.65
ARGM-MNR-H0       0       0       2     0.00    0.00    0.00
  ARGM-NEG       17       2      10    89.47   62.96   73.91
  ARGM-PNC        2       5      10    28.57   16.67   21.05
  ARGM-TMP      302      65     108    82.29   73.66   77.73
    R-ARG0        7       9      30    43.75   18.92   26.42
    R-ARG1        7      13      18    35.00   28.00   31.11
    R-ARG2        2       1      11    66.67   15.38   25.00
    R-ARG3        0       0       3     0.00    0.00    0.00
R-ARGM-TMP        0       0       1     0.00    0.00    0.00
   Support      541     372     441    59.26   55.09   57.10
------------------------------------------------------------
```

Table 4.7: Detailed score of the best combined identification and classification on test section 23

Chapter 5

Future Work

5.1 Further improving PropBank and NomBank SRL

5.1.1 Improving PropBank SRL

Experimental results in Section 3.4 of Chapter 3 show the significance of argument ordering during classification. An optimal ordering should put individual arguments into the most relevant and discriminative context. We are still experimenting with other ordering possibilities, as well as context features that do not depend on ordering.

Chapter 3 focused on semantic argument classification, assuming correct argument identification. A natural extension is to thoroughly consider the semantic context during argument identification, so as to more tightly integrate argument identification and classification.

Besides basic semantic context features used in Chapter 3, more intricate ones have been experimented with promising results. For instance, "argument path" consisting of the syntactic path from one argument to another neighboring argument, has shown

to be effective in classification. We believe that more semantic context features are available from the full syntactic parse tree and underlying argument elements.

Semantic context features may even go beyond the predicate level if we consider all the context provided by multiple predicates within a single sentence.

It is also worthwhile to design a more informed Voting algorithm with each vote weighted by a prior based on the voting classifier's accuracy. We can also try to invoke voting during the process of Beam searching the best global argument sequence, since classifiers based on the *Ar* context feature make very different errors from the others.

The Sigmoid function we used to convert SVM output to a confidence value is empirically simple and effective, but [Platt, 1999] and [Zadrozny and Elkan, 2002] suggested methods to better fit binary and multi-class SVM output into probability estimates using complex learning algorithms. More experiments are necessary to explore their effects in voting and beam search.

5.1.2 Integrating PropBank and NomBank SRL

Work in [Pustejovsky *et al.*, 2005] discussed the possibility of merging various Treebank annotation efforts including PropBank, NomBank, and others. We are currently studying ways of concurrently producing automatic PropBank and NomBank SRL, and improving the accuracy by exploiting the inter-relationship between verb predicate-argument and noun predicate-argument structures.

Besides the obvious correspondence between a verb and its nominalizations, e.g., "replace" and "replacement", there is also correspondence between verb predicates in PropBank and support verbs in NomBank. Statistics from NomBank sections 02-21 show that 86% of the support verbs in NomBank are also predicate verbs in PropBank. When they coincide, they share 18,250 arguments of which 63% have the same argument class in PropBank and NomBank.

The approaches under investigation include:

- Using PropBank data as augmentation to NomBank training data.
- Using re-ranking techniques [Collins, 2000] to jointly improve Probank and NomBank SRL accuracy.

5.1.3 Integrating Syntactic and Semantic Parsing

The analysis and synthesis of natural language can be approached from both the syntactic and the semantic aspect. Previous work has extensively studied the possibility of using syntactic structures to infer meanings, exemplified by SRL based on deep or shallow syntactic structures. The other direction of using grammatical components' semantic categories to infer their syntactic structures is less well studied. There has been unsuccessful attempts at improving syntactic parsing through re-ranking of semantically labelled parse trees [Gildea and Jurafsky, 2002; Sutton and McCallum, 2005].

Unlike the re-ranking approach, we propose to add semantically motivated features to a statistical syntactic parser. The aim is to build a more tightly integrated language analyzer that iteratively or concurrently use syntactic and semantic information. An analogy is a foreign language learner, who starts with simple words with concrete meanings, then iteratively learns more grammar (syntax) and more vocabulary (semantics), until the skill of analyzing and producing complex sentences is acquired.

5.2 Applications of SRL

5.2.1 SRL based Question Answering

Many current successful automatic Question Answering (QA) systems are based on syntactic analysis of questions and answer source text passages. Question answering becomes statistical analysis of string chunks. World knowledge is often introduced in the form of various taxonomies and dictionaries, but the problem is still not tackled directly from a semantic point of view. Ideally, QA should be solved by trying to understand the meaning and to associate question and answer semantic components, instead of string chunks. There has been research in using FrameNet annotation to construct templates used in Question Answering system [Pradhan *et al.*, 2002].

We experimented with a QA system integrated with a PropBank based semantic parser. Preliminary results [Chen *et al.*, 2004] are far from the state-of-the-art in terms of QA accuracy, but are interesting and can serve as a starting point of further research.

Chapter 6

Conclusion

This thesis has discussed automatic semantic analysis of natural langauge sentences, using machine learning techniques applied to the PropBank and NomBank corpus. Particularly, it considers the following two tasks:

- How do arguments in a predicate-argument structure depend on each other, and how to exploit their interdependence to improve PropBank based SRL system?
- How to make use of the newly available NomBank to build an SRL system that produces nominal predicate-argument structures?

Chapter 3 reveals the close interdependence among arguments of a verb predicate, using concrete examples and empirical studies. It also demonstrates the deficiency of SRL systems which label each semantic argument independently. Systematic experiments and analysis of results prove the effectiveness of the proposed SRL system. Evaluation on the standard WSJ test section 23 shows that an augmented SRL system using *context features* as proposed significantly improve over a baseline system in the argument classification task.

Chapter 4 documents the empirical study of how a NomBank based SRL system can be implemented based on previous experience with PropBank SRL systems. A

large feature set previously shown effective in PropBank SRL task is adapted to the NomBank SRL context. An additional NomBank specific feature set is proposed. A subset of these features is chosen by a greedy feature selection algorithm. The additional NomBank specific features experimented significantly improves the argument classification task.

The contributions of this thesis are:

- Proposing an effective set of *semantic context* features. Emphasizing the importance of capturing argument interdependence in building automatic SRL systems.
- Providing one of the first known NomBank based SRL systems.

The work presented here serves as the basis for further investigation. The ultimate goal should be to accurately analyze sentences' semantic structures, both for the verb predicates and for the nominal predicates. Such ability will be one of the keys to more powerful natural langauge processing applications.

Bibliography

[Baker *et al.*, 1998] C. F. Baker, C. J. Fillmore, and J. B. Lowe. The Berkeley FrameNet Project. In *Proceedings of COLING-1998*, 1998.

[Bejan *et al.*, 2004] Cosmin Adrian Bejan, Alessandro Moschitti, Paul Morarescu, Gabriel Nicolae, and Sanda Harabagiu. Semantic Parsing Based on FrameNet. In *Proceedings of SENSEVAL-3*, 2004.

[Berger *et al.*, 1996] Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics, (22-1), March 1996*, 1996.

[Carreras and Marquez, 2004] Xavier Carreras and Lluis Marquez. Introduction to the CoNLL-2004 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL-2004*, 2004.

[Carreras and Marquez, 2005] Xavier Carreras and Lluis Marquez. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL-2005*, 2005.

[Charniak and Johnson, 2005] Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of ACL-2005*, 2005.

[Charniak, 2000] Eugene Charniak. A Maximum Entropy Inspired Parser. In *Proceedings of NAACL-2000*, 2000.

[Chen *et al.*, 2004] Cao Chen, Wei Wei Chen, and Zheng Ping Jiang. Applying Semantic Parsing to Question Answering System, 2004.

[Collins, 1999] Michael Collins. *Head-driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.

[Collins, 2000] Michael Collins. Discriminative Reranking for Natural Language Parsing. In *Proc. 17th International Conf. on Machine Learning*, 2000.

[Ellsworth *et al.*, 2004] Michael Ellsworth, Katrin Erk, Paul Kingsbury, and Sebastian Pado. PropBank, SALSA, and FrameNet: How Design Determines Product. In *Proceedings of the LREC 2004 Workshop on Building Lexical Resources from Semantically Annotated Corpora*, 2004.

[Fleischman *et al.*, 2003] Michael Fleischman, Namhee Kwon, and Eduard Hovy. Maximum Entropy Models for FrameNet Classification. In *Proceedings of EMNLP-2003*, 2003.

[Gildea and Jurafsky, 2002] Daniel Gildea and Daniel Jurafsky. Automatic Labeling of Semantic Roles. *Computational Linguistics*, 2002.

[Hacioglu *et al.*, 2004] Kadri Hacioglu, Sameer Pradhan, Wayne Ward, James H.Martin, and Daniel Jurafsky. Semantic Role Labeling by Tagging Syntactic Chunks. In *Proceedings of CoNLL-2004*, 2004.

[Haghighi *et al.*, 2005] Aria Haghighi, Kristina Toutanova, and Christopher D. Manning. A Joint Model for Semantic Role Labeling. In *Proccedings of CoNLL-2005*, 2005.

[Jiang and Ng, 2006] Zheng Ping Jiang and Hwee Tou Ng. Semantic Role Labeling of NomBank: A Maximum Entropy Approach. In *2006 Conference on Empirical Methods in Natural Language Processing, to appear*, 2006.

[Jiang *et al.*, 2005] Zheng Ping Jiang, Jia Li, and Hwee Tou Ng. Semantic Argument Classification Exploiting Argument Interdependence. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, 2005.

[Kingsbury and Palmer, 2003] Paul Kingsbury and Martha Palmer. PropBank: the Next Level of TreeBank. In *Proceedings of Treebanks and Lexical Theories*, 2003.

[Kingsbury *et al.*, 2002] Paul Kingsbury, Martha Palmer, and Mitch Marcus. Adding Semantic annotation to the Penn Treebank. In *Proceedings of HLT-2002*, 2002.

[Kingsbury *et al.*, 2003] Paul Kingsbury, Benjamin Snyder, Nianwen Xue, and Martha Palmer. PropBank as a Bootstrap for Richer Annotation Schemes. In *The 6th Workshop on Interlinguas: Annotations and Translations*, 2003.

[Kouchnir, 2004] Beata Kouchnir. A Memory-based Approach for Semantic Role Labeling. In *Proceedings of CoNLL-2004*, 2004.

[Kudo and Matsumoto, 2001] Taku Kudo and Yuji Matsumoto. Chunking with Support Vector Machines. In *Proceedings of NAACL-2001*, 2001.

[Kwon *et al.*, 2004] Namhee Kwon, Michael Fleischman, and Eduard Hovy. FrameNet-based Semantic Parsing using Maximum Entropy Models. In *Proceedings of COLING-2004*, 2004.

[Lim *et al.*, 2004] Joon-Ho Lim, Young-Sook Hwang, So-Young Park, and Hae-Chang Rim. Semantic Role Labeling using Maximum Entropy Model. In *Proceedings of CoNLL-2004*, 2004.

[Litkowski, 2004] Kenneth C. Litkowski. SENSEVAL-3 Task Automatic Labeling of Semantic Roles. In *Proceedings of Senseval-3: The Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, ACL*, 2004.

[Macleod *et al.*, 1997] Catherine Macleod, Adam Meyers, Ralph Grishman, Leslie Barrett, and Ruth Reeves. Designing a Dictionary of Derived Nominals. In *Proceedings of Recent Advances in Natural Language Processing*, 1997.

[Macleod *et al.*, 1998] Catherine Macleod, Ralph Grishman, Adam Meyers, Leslie Barrett, and Ruth Reeves. NOMLEX: A Lexicon of Nominalizations. In *Proceedings of EURALEX'98*, 1998.

[Marcus *et al.*, 1993] Mitch Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 1993.

[Marcus, 1994] Mitch Marcus. The Penn TreeBank: A revised corpus design for extracting predicate-argument structure. In *Proceedings of the ARPA Human Language Technology Workshop*, 1994.

[Marquez *et al.*, 2005] Lluis Marquez, Pere Comas, Jusus Gimenez, and Neus Catala. Semantic Role Labeling as Sequential Tagging. In *Proceedings of CoNLL-2005*, 2005.

[Meyers *et al.*, 2004a] Adam Meyers, Ruth Reeves, and Catherine Macleod. NP-External Arguments: A Study of Argument Sharing in English. In *The ACL 2004 Workshop on Multiword Expressions: Integrating Processing*, 2004.

[Meyers *et al.*, 2004b] Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekeley, Veronkia Zielinska, and Brian Young. The Cross-Breeding of Dictionaries. In *Proceedings of LREC-2004*, 2004.

[Meyers *et al.*, 2004c] Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. Annotating Noun Argument Structure for NomBank. In *Proceedings of LREC-2004*, 2004.

[Meyers *et al.*, 2004d] Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronkia Zielinska, Brian Young, and Ralph Grishman. The NomBank Project: An Interim Report. In *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, 2004.

[Moldovan *et al.*, 2004] Dan Moldovan, Roxana Girju, Marian Olteanu, and Ovidiu Fortu. SVM Classification of FrameNet Semantic Roles. In *Proceedings of SENSEVAL-3*, 2004.

[Palmer and Marcus, 2002] Martha Palmer and Mitch Marcus. *PropBank annotation guidelines*, 2002.

[Palmer *et al.*, 2005] Martha Palmer, Paul Kingsbury, and Daniel Gildea. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 2005.

[Park *et al.*, 2004] Kyung-Mi Park, Young-Sook Hwang, and Hae-Chang Rim. Two-Phase Semantic Role Labeling based on Support Vector Machines. In *Proceedings of CoNLL-2004*, 2004.

[Platt, 1999] John C. Platt. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In *Advances in Large Margin Classifiers*, 1999.

[Pradhan *et al.*, 2002] Sameer Pradhan, Valerie Krugler, Wayne Ward, Daniel Jurafsky, and James H. Martin. Using Semantic Representations in Question Answering. In *Proceedings of ICON-2002*, 2002.

[Pradhan *et al.*, 2004] Sameer S. Pradhan, Honglin Sun, Wayne Ward, James H. Martin, and Dan Jurafsky. Parsing Arguments of Nominalizations in English and Chinese. In *Proceedings of HLT/NAACL 2004*, 2004.

[Pradhan *et al.*, 2005a] Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H. Martin, and Daniel Jurafsky. Semantic Role Chunking Combing Complementary Syntactic Views. 2005.

[Pradhan *et al.*, 2005b] Sameer Pradhan, Valerie Krugler, Wayne Ward, James H. Martin, and Daniel Jurafsky. Support Vector Learning for Semantic Argument Classification. *Machine Learning Journal*, 2005.

[Punyakanok *et al.*, 2004] Vasin Punyakanok, Dan Roth, Wen-Tau Yih, and Dav Zimak. Semantic Role Labeling via Integer Linear Programming Inference. In *Proceedings of COLING-2004*, 2004.

[Punyakanok *et al.*, 2005] Vasin Punyakanok, Peter Koomen, Dan Roth, and Wen Tau Yih. Generalized Inference with Multiple Semantic Role Labeling Systems. In *Proceedings of CoNLL-2005*, 2005.

[Pustejovsky *et al.*, 2005] James Pustejovsky, Adam Meyers, Martha Palmer, and Massimo Poesio. Merging PropBank, NomBank, TimeBank, Penn Discourse Treebank and Coreference. In *Workshop Frontiers in Corpus Annotation II: Pie in the Sky, ACL 2005*, 2005.

[Ratnaparkhi, 1998] Adwait Ratnaparkhi. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. PhD thesis, University of Pennsylvania, 1998.

[Sang *et al.*, 2005] Erik Tjong Kim Sang, Sander Canisius, Antal van den Bosch, and Toine Bogers. Applying Spelling Error Correction Techniques for Improving Semantic Role Labeling. In *Proceedings of CoNLL-2005*, 2005.

[Sidney and Castellan Jr., 1988] Siegel Sidney and N.John Castellan Jr. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill, 1988.

[Sutton and McCallum, 2005] Charles Sutton and Andrew McCallum. Joint Parsing and Semantic Role Labeling. In *Proceedings of CoNLL-2005*, 2005.

[Thompson *et al.*, 2004] Cynthia A. Thompson, Siddharth Patwardhn, and Carolin Arnold. Generative Models for Semantic Role Labeling. In *Proceedings of SENSEVAL-3*, 2004.

[Toutanova *et al.*, 2005] Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. Joint Learning Improves Semantic Role Labeling. In *Proceedings of ACL-2005*, 2005.

[Xue and Palmer, 2004] Nianwen Xue and Martha Palmer. Calibrating Features for Semantic Role Labeling. In *Proceedings of EMNLP-2004*, 2004.

[Zadrozny and Elkan, 2002] Bianca Zadrozny and Charles Elkan. Transforming Classifier Scores into Accurate Multiclass Probability Estimates. In *Proceedings of KDD-2002*, 2002.