

Robust Semantic Role Labeling

by

Sameer. S. Pradhan

B.E., University of Bombay, 1994

M.S., Alfred University, 1997

A thesis submitted to the
Faculty of the Graduate School of the
University of Colorado in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Computer Science

2006

This thesis entitled:
Robust Semantic Role Labeling
written by Sameer. S. Pradhan
has been approved for the Department of Computer Science

Prof. Wayne Ward

Prof. James Martin

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Pradhan, Sameer. S. (Ph.D., Computer Science)

Robust Semantic Role Labeling

Thesis directed by Prof. Wayne Ward

The natural language processing community has recently experienced a growth of interest in domain independent semantic role labeling. The process of semantic role labeling entails identifying all the predicates in a sentence, and then, identifying and classifying sets of word sequences, that represent the arguments (or, semantic roles) of each of these predicates. In other words, this is the process of assigning a *WHO* did *WHAT* to *WHOM*, *WHEN*, *WHERE*, *WHY*, *HOW* etc. structure to plain text, so as to facilitate enhancements to algorithms that deal with various higher-level natural language processing tasks, such as – information extraction, question answering, summarization, machine translation, etc., by providing them with a layer of semantic structure on top of the syntactic structure that they currently have access to. In recent years, there have been a few attempts at creating hand-tagged corpora that encode such information. Two such corpora are FrameNet and PropBank. One idea behind creating these corpora was to make it possible for the community at large, to train supervised machine learning classifiers that can be used to automatically tag vast amount of unseen text with such shallow semantic information. There are various types of predicates, the most common being verb predicates and noun predicates. Most work prior to this thesis was focused on arguments of verb predicates. This thesis primarily addresses three issues: i) improving performance on the standard data sets, on which others have previously reported results, by using a better machine learning strategy and by incorporating novel features, ii) extending this work to parse arguments of nominal predicates, which also play an important role in conveying the semantics of a passage, and iii) investigating methods to improve the robustness of the classifier across different genre of text.

Dedication

To *Aai* (mother), *Baba* (father) and *Dada* (brother)

Acknowledgements

There are several people in different circles of life that have contributed towards my successfully finishing this thesis. I will try to thank each one of them in the logical group that they represent. Since there are so many different people who were involved, I might miss a few names. If you are one of them, please forgive me for that, and consider it to be a failure on part of my mental retentive capabilities.

First and foremost comes my family – I would like to thank my wonderful parents and my brother, for cultivating the importance of higher education in me. They somehow managed, though initially, with great difficulties, to inculcate an undying thirst for knowledge inside me, and provided me with all the necessary encouragement and motivation which made it possible for me to make an attempt at expressing my gratitude through this acknowledgment today.

Second come the mentors – I would like to thank my advisors – professors Wayne Ward , James Martin and Daniel Jurafsky – especially Wayne, and Jim who could not escape my incessant torture – both in and out of the office, taking it all in with a smiling face, and giving me the most wonderful advice and support with a little chiding at times when my behavior was unjustified, or calming me down when I worried too much about something that did not matter in the long run. Dan somehow got lucky and did not suffer as much since he moved away to Stanford in 2003, but he did receive his share. Initially, professor Martha Palmer from the University of Pennsylvania, played a more external role, but a very important one, as almost all the experiments in this thesis are

performed on the PropBank database that was developed by her. Without that data, this thesis would not be possible. In early 2004, she graciously agreed to serve on my thesis committee, and started playing a more active role as one of my advisors. It was quite a coincidence that by the time I defended my thesis, she was a part of the faculty at Boulder. Greg Grudic was a perfect complement to the committee because of his core interests in machine learning, and provided few very crucial suggestions that improved the quality of the algorithms. Part of the data that I also experimented with and which complemented the PropBank data was FrameNet. For that I would like to thank professors Charles Fillmore, Collin Baker, and Srinu Narayanan from the International Computer Science Institute (ICSI), Berkeley. Another person that played a critical role as my mentor, but who was never really part of the direct thesis advisory committee, was professor Ronald Cole. I know people who get sick and tired of their advisors, and are glad to graduate and move away from them. My advisors were so wonderful, that I never felt like graduating. When the time was right, they managed to help me make my transition out of graduate school.

Third comes the thanks to money. The funding organizations – without which all the earlier support and guidance would have never come to fruition. At the very beginning, I had to find someone to fund my education, and then organizations to fund my research. If it wasn't for Jim's recommendation to meet Ron – back in 2000 when I was in serious academic turmoil – to seek for any funding opportunity, I would not have been writing this today. This was the first time I met Ron and Wayne. They agreed to give me a summer internship at the Center for Spoken Language Research (CSLR), and hoped that I could join the graduate school in the Fall of 2000, if things were conducive. At the end of that summer, thanks to an email by Ron, and recommendations from him and Wayne to admit me as a graduate student in the Computer Science Department, to Harold Gabow, who was then the Graduate Admissions Coordinator, accompanied by their willingness to provide financial support for my PhD, the latter put my admission

process in high gear, and I was admitted to the PhD program at Colorado. Although CSLR was mainly focused on research in speech processing, my research interests in text processing were also shared by Wayne, Jim and Dan, who decided to collaborate with Kathleen McKeown and Vasileios Hatzivassiloglou at Columbia University, and apply for a grant from the ARDA AQUAINT program. Almost all of my thesis work has been supported by this grant via contract OCG4423B. Part of the funding also came from the NSF via grants IS-9978025 and ITR/HCI 0086132.

Then come the faithful machines. My work was so much computation intensive, that I was always hungry for machines. I first grabbed all the machines I could muster at CSLR. Some of which were part of a grant from Intel, and some which were procured from the aforementioned grants. When research was in its peak, and existing machinery was not able to provide the required CPU cycles, I also raided two clusters of machines from professor Henry Tufo – The “Hemisphere” cluster and the “Occam” cluster. This hardware was in turn provided by NSF ARI grant CDA-9601817, NSF MRI grant CNS-0420873, NASA AIST grant NAG2-1646, DOE SciDAC grant DE-FG02-04ER63870, NSF sponsorship of the National Center for Atmospheric Research, and a grant from the IBM Shared University Research (SUR) program. Without the faithful work undertaken by these machines, it would have taken me another four to five years to generate the state-of-the-art, cutting-edge, performance numbers that went in this thesis – which by then, would not have remained state-of-the-art. There were various people I owe for the support they gave in order to make these machine available day and night. Most important among them were Matthew Woitaszek, Theron Voran, Michael Oberg, and Jason Cope.

Then the researchers and students at CSLR and CU as a whole with whom I had many helpful discussions that I found extremely enlightening at times. They were Andy Hagen, Ayako Ikeno, Bryan Pellom, Kadri Hacioglu, Johannes Henkel, Murat Akbacak, and Noah Coccaro.

Then my social circle in Boulder. The friends without whom existence in Boulder would have been quite a drab, and maybe I might have wanted to actually graduate prematurely. Among them were Rahul Patil, Mandar Rahurkar, Rahul Dabane, Gautam Apte, Anmol Seth, Holly Krech, Benjamin Thomas. Here I am sure I am forgetting some more names. All of these people made life in Boulder an enriching experience.

Finally, comes the academic community in general. Outside the home and university and friend circle, there were, then, some completely foreign personalities with whom I had secondary connections – through my advisors, and of whom some happen to be not so completely foreign anymore, gave a helping hand. Of them were Ralph Weischedel and Scott Miller from BBN Technologies, who let me use their named entity tagger – *IdentiFinder*; Dan Gildea for providing me with a lot of initial support and his thesis which provided the ignition required to propel me in this area of research. Julia Hockenmaier provided me with the the gold standard CCG parser information which was invaluable for some experiments.

Contents

Chapter	
1	Introduction 1
2	History of Computational Semantics 6
2.1	The Semantics View 6
2.2	The Computational View 16
2.2.1	BASEBALL 16
2.2.2	ELIZA 17
2.2.3	SHRDLU 19
2.2.4	LUNAR 19
2.2.5	NLPQ 20
2.2.6	MARGIE 20
2.3	Early Semantic Role Labeling Systems 24
2.4	Advent of Semantic Corpora 25
2.5	Corpus-based Semantic Role Labeling 28
2.5.1	Problem Description 30
2.6	The First Cut 30
2.7	The First Wave 33
2.7.1	The Gildea and Palmer (G&P) System 34
2.7.2	The Surdeanu <i>et al.</i> System 34

2.7.3	The Gildea and Hockenmaier (G&H) System	35
2.7.4	The Chen and Rambow (C&R) System	36
2.7.5	The Flieschman <i>et al.</i> System	37
3	Automatic Statistical SEMantic Role Tagger – ASSERT	38
3.1	ASSERT Baseline	38
3.1.1	Classifier	39
3.1.2	System Implementation	40
3.1.3	Baseline System Performance	41
3.2	Improvements to ASSERT	42
3.2.1	Feature Engineering	42
3.2.2	Feature Selection and Calibration	51
3.2.3	Disallowing Overlaps	52
3.2.4	Argument Sequence Information	54
3.2.5	Alternative Pruning Strategies	55
3.2.6	Best System Performance	57
3.2.7	System Analysis	58
3.2.8	Comparing Performance with Other Systems	63
3.2.9	Using Automatically Generated Parses	65
3.2.10	Comparing Performance with Other Systems	66
3.3	Labeling Text from a Different Corpus	69
3.3.1	AQUAINT Test Set	69
4	Arguments of Nominalizations	71
4.1	Introduction	71
4.2	Semantic Annotation and Corpora	72
4.3	Baseline System	73
4.4	New Features	74

4.5	Best System Performance	76
4.6	Feature Analysis	76
4.7	Discussion	77
5	Different Syntactic Views	78
5.1	Baseline System	79
5.2	Alternative Syntactic Views	81
5.2.1	Minipar-based Semantic Labeler	81
5.2.2	Chunk-based Semantic Labeler	85
5.3	Combining Semantic Labelers	87
5.4	Improved Architecture	90
5.5	System Description	91
5.6	Results	93
6	Robustness Experiments	95
6.1	The Brown Corpus	96
6.2	Semantic Annotation	97
6.3	Experiments	98
6.3.1	Experiment 1: How does ASSERT trained on WSJ perform on Brown?	98
6.3.2	Experiment 2: How well do the features transfer to a different genre?	100
6.3.3	Experiment 3: How much does correct structure help?	102
6.3.4	Experiment 4: How sensitive is semantic argument prediction to the syntactic correctness across genre?	103
6.3.5	Experiment 5: How much does combining syntactic views help overcome the errors?	106
6.3.6	Experiment 6: How much data do we need to adapt to a new genre?	107

7	Conclusions and Future Work	109
7.1	Summary of Experiments	109
7.1.1	Performance Using Correct Syntactic Parses	109
7.1.2	Using Output from a Syntactic Parser	110
7.1.3	Combining Syntactic Views	111
7.2	What does it mean to be correct?	112
7.3	Robustness to Genre of Data	112
7.4	General Discussion	114
7.5	Nominal Predicates	117
7.6	Considerations for Corpora	117
	Bibliography	119
	Appendix	
A	Temporal Words	127

Tables

Table

2.1	Conceptual-dependency primitives.	22
2.2	Argument labels associated with the predicate <i>operate</i> (sense: work) in the PropBank corpus.	28
2.3	Argument labels associated with the predicate <i>author</i> (sense: to write or construct) in the PropBank corpus.	29
2.4	List of adjunctive arguments in PropBank – ARGMS	30
2.5	Distributions used for semantic argument classification, calculated from the features extracted from a Charniak parse.	33
3.1	Baseline performance on all the three tasks using “gold-standard” parses	42
3.2	Argument labels associated with the two senses of predicate <i>talk</i> in PropBank corpus.	45
3.3	Performance improvement on selecting features per argument and calibrating the probabilities on 10k training data.	52
3.4	Improvements on the task of argument identification and classification after disallowing overlapping constituents.	54
3.5	Improvements on the task of argument identification and classification using Treebank parses, after performing a search through the argument lattice.	54
3.6	Comparing pruning strategies	56

3.7	Best system performance on all three tasks using Treebank parses. . . .	57
3.8	Best system performance on all three tasks on the latest PropBank data, using Treebank parses.	57
3.9	Effect of each feature on the argument classification task and argument identification task, when added to the baseline system.	60
3.10	Improvement in classification accuracies after adding named entity infor- mation.	60
3.11	Performance of various feature combinations on the task of argument classification.	61
3.12	Performance of various feature combinations on the task of argument identification	62
3.13	Precision/Recall table for the combined task of argument identification and classification using Treebank parses.	63
3.14	Argument classification using same features but different classifiers. . . .	64
3.15	Argument identification	64
3.16	Argument classification	65
3.17	Argument Identification and classification	65
3.18	Performance degradation when using automatic parses instead of Tree- bank ones.	66
3.19	Best system performance on all tasks using automatically generated syn- tactic parses.	67
3.20	Argument identification	68
3.21	Argument classification	68
3.22	Argument Identification and classification	68
3.23	Performance on the AQUAINT test set.	69
3.24	Feature Coverage on PropBank test set using semantic role labeler trained on PropBank training set.	70

3.25	Coverage of features on AQUAINT test set using semantic role labeler trained on PropBank training set.	70
4.1	Baseline performance on all three tasks.	73
4.2	Best performance on all three tasks.	76
5.1	Features used in the Baseline system	80
5.2	Baseline system performance on all tasks using Treebank parses and automatic parses on PropBank data.	80
5.3	Features used in the Baseline system using Minipar parses.	84
5.4	Baseline system performance on all tasks using Minipar parses.	84
5.5	Head-word based performance using Charniak and Minipar parses.	84
5.6	Features used by chunk based classifier.	86
5.7	Semantic chunker performance on the combined task of Id. and classification.	86
5.8	Constituent-based best system performance on argument identification and argument identification and classification tasks after combining all three semantic parses.	87
5.9	Performance improvement on parses changed during pair-wise Charniak and Chunk combination.	88
5.10	Performance improvement on head word based scoring after oracle combination. Charniak (C), Minipar (M) and Chunker (CH).	89
5.11	Performance improvement on head word based scoring after combination. Charniak (C), Minipar (M) and Chunker (CH).	89
5.12	Performance of the integrated architecture on the CoNLL-2005 shared task on semantic role labeling.	94

6.1	Number of predicates that have been tagged in the PropBanked portion of Brown corpus	97
6.2	Performance on the entire PropBanked Brown corpus.	99
6.3	Constituent deletions in WSJ test set and the entire PropBanked Brown corpus.	100
6.4	Performance when ASSERT is trained using correct Treebank parses, and is used to classify test set from either the same genre or another. For each dataset, the number of examples used for training are shown in parenthesis	102
6.5	Performance of different versions of Charniak parser used in the experiments.	103
6.6	Performance on WSJ and Brown test set when ASSERT is trained on features extracted from automatically generated syntactic parses	105
6.7	Performance of the task of argument identification and classification using architecture that combines top down syntactic parses with flat syntactic chunks.	106
6.8	Effect of incrementally adding data from a new genre	108

Figures

Figure

1.1	PropBank example.	2
1.2	5
2.1	Standard Theory.	12
2.2	Extended Standard Theory.	13
2.3	Example database in BASEBALL	17
2.4	Example conversation with ELIZA.	18
2.5	Example interaction in SHRDLU	19
2.6	Schank’s conceptual cases.	21
2.7	Conceptual-dependency representation of “The big boy gives apples to the pig.”	22
2.8	FrameNet example.	26
2.9	Syntax tree for a sentence illustrating the PropBank tags	28
2.10	A sample sentence from the PropBank corpus	31
2.11	Illustration of path NP↑S↓VP↓VBD	32
2.12	List of content word heuristics.	35
3.1	The semantic role labeling algorithm	41
3.2	Noun head of PP.	45
3.3	Ordinal constituent position.	46

3.4	Tree distance.	47
3.5	Sibling features.	47
3.6	CCG parse.	48
3.7	Plots showing true probabilities versus predicted probabilities before and after calibration on the test set for ARGUMENT-TMP	53
3.8	Learning curve for the task of identifying and classifying arguments using Treebank parses.	61
4.1	Nominal example.	72
5.1	Illustration of how a parse error affects argument identification.	82
5.2	PSG and Minipar views.	83
5.3	Semantic Chunker.	85
5.4	New Architecture.	91
5.5	Example classification using the new architecture.	93

Chapter 1

Introduction

The proliferation of on-line natural language text on a wide variety of topics, coupled with the recent improvements in computer hardware, presents a huge opportunity for the natural language processing community. Unfortunately, while a considerable amount of information can be extracted from raw text (Sebastiani, 2002), much of the latent semantic content in these collections remains beyond the reach of current systems. It would be easier to extract information from these collections if they are given some form of higher level semantic structure automatically. This can play a key role in natural language processing applications such as information extraction, question answering, summarization, machine translation, etc.

Semantic role labeling is a technique that has great potential for achieving this goal. When presented with a sentence, the labeler should find all the predicates in that sentence, and for each predicate, identify and label its semantic arguments. This process entails identifying sets of word sequences in the sentence that represent these semantic arguments, and assigning specific labels to them. This notion of semantic role, or case role, or thematic role analysis has a long history in the computational linguistics literature. We look more into their history in Chapter 2

An example could help clarify the notion of semantic role labeling. Consider the

following sentence:

1. Bell Atlantic Corp. said it will acquire one of Control Data Corp.'s computer maintenance businesses

In this sentence, there are two verb predicates – *said* and *acquire*. For the predicate *said*, the entity that is the *sayer*, or the WHO part of the representation, is *Bell Atlantic Corp.* whereas, *it will acquire one of Control Data Corp.'s computer maintenance businesses* is *the thing being said*, or the WHAT part. Upon identifying all the possible roles or arguments of both the predicates, the sentence can be represented as in the Figure 1.1. In addition to these roles, there could be others that represent WHEN, WHERE, HOW, etc., as shown.

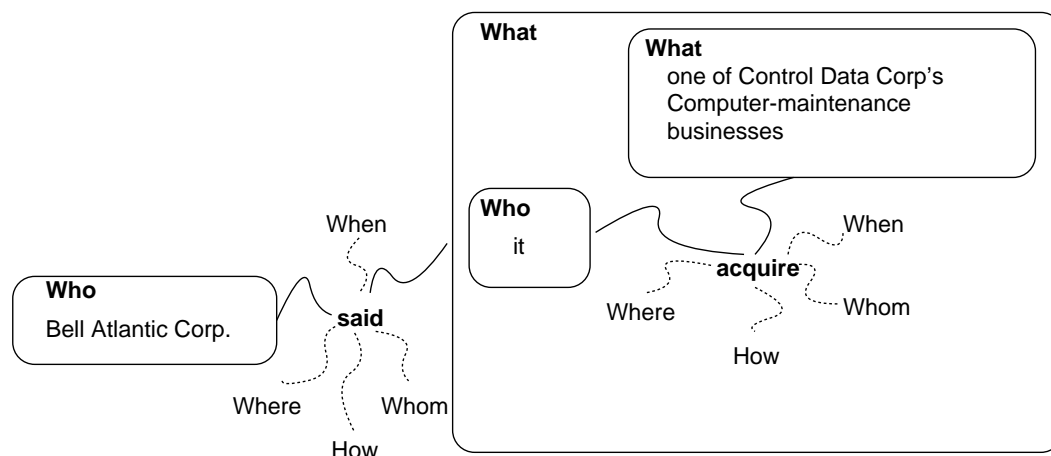


Figure 1.1: PropBank example.

Let us now take a look at how semantic role labeling, coupled with inference, can be used to answer questions that would otherwise be difficult to answer using a purely word-based mechanism. Figure 1.2 shows the semantic analysis of a question “Who assassinated President McKinley?”, along top few false positives ranked by a question-answering system, and then the correct answer. The ranks of the documents are mentioned alongside.

For this question a slightly sophisticated question answering system (Pradhan et al., 2002) which incorporated some level of semantic information, by adding heuristics identifying named entities that the answer would tend to be instantiated as, identified *Theodore Roosevelt* as being the assassin. In this case, the system knew that the answer to this question would be a PERSON named entity. However, since it did not have any other semantic information, for example, the predicate-argument structure, it succumbed to an error. This could have been avoided if the system had the knowledge that *Theodore Roosevelt* was not the *assassin* of *William McKinley*, or in other words, *Theodore Roosevelt* did not play the WHO role of the predicate *assassinated*. Such semantic representation could also help summarize/formulate answers that are not available as sub-strings of the text from which the answer is sought.

There has been a resurgence of interest in systems that create such shallow analysis in recent years due to the convergence of a number of related factors: recent successes in the creation of robust wide-coverage syntactic parsers (Charniak, 2000), breakthroughs in the use of robust supervised machine learning approaches, such as support vector machines, on natural language data, and last, but not the least, the availability of large amounts of relevant human-labeled data. There is a growing consensus in the field that algorithms need to be designed to produce robust and accurate shallow semantic analysis for a wide variety of naturally occurring texts. These texts can be of various different types – newswire, usenet, broadcast news transcripts, conversations, etc. In short, the texts can range from one that are grammatically well-formed, to ones that not only lacks grammatical coherence, but also consists of disfluencies. As we shall see, our approach relies heavily on supervised machine learning techniques, and so we focus on the genre of text that has hand-tagged data available for learning. The presence of syntactically tagged newswire data has enabled the creation of high-quality syntactic parsers.

Leveraging this information, researchers have started semantic role analysis on the same genre of text, and that, in fact will be the focus of this thesis. It will investigate into new features and better learning mechanisms that would improve the accuracy of the current state-of-the-art in semantic role labeling. It will try to increase the robustness of the system so that not only does it perform well on the genre of text that it is trained on, but it will degrade gracefully to a new text source. Finally, the state-of-the-art parser thus developed is being made freely available to the NLP community, and it is envisioned that this will drive further research in the field of text understanding.

<p><i>Question:</i> Who assassinated President McKinley?</p> <p><i>Parse:</i> $[role=agent \text{ Who}] [target \text{ assassinated}] [role=theme [ne=person_description \text{ President}]] [ne=person \text{ McKinley}]$?</p> <p><i>Keywords:</i> assassinated President McKinley</p> <p><i>Answer named entity (ne) Type:</i> Person</p> <p><i>Answer thematic role (role) Type:</i> Agent of target synonymous with “assassinated”</p> <p><i>Thematic role pattern:</i> $[role=agent [ne=person \text{ ANSWER}]] \wedge [target \text{ synonym_of}(\text{assassinated})] \wedge [role=theme [ne=person \text{ reference_to}(\text{President McKinley})]]$</p> <p>This is one of possibly more than one patterns that will be applied to the answer candidates.</p>
<p><i>False Positives:</i></p> <p>Note: The sentence number indicates the final rank of that sentence in the returns using named entity information for re-ranking. no thematic role information was used.</p> <ol style="list-style-type: none"> In $[ne=date \text{ 1904}]$, $[ne=person_description \text{ President}]$ $[ne=person \text{ Theodore Roosevelt}]$, who had succeeded the <i>assassinated</i> $[ne=person \text{ William McKinley}]$, was elected to a term in his own right as he defeated $[ne=person_description \text{ Democrat}]$ $[ne=person \text{ Alton B. Parker}]$. $[ne=person \text{ Hanna}]$'s worst fears were realized when $[ne=person_description \text{ President}]$ $[ne=person \text{ William McKinley}]$ was <i>assassinated</i>, but the country did rather well under TR's leadership anyway. $[ne=person \text{ Roosevelt}]$ became president after $[ne=person \text{ William McKinley}]$ was <i>assassinated</i> in $[ne=date \text{ 1901}]$ and served until $[ne=date \text{ 1909}]$.
<p><i>Correct Answer:</i></p> <ol style="list-style-type: none"> $[role=temporal \text{ In } [ne=date \text{ 1901}]]$, $[role=theme [ne=person_description \text{ President}]]$ $[ne=person \text{ William McKinley}]$ was $[target \text{ shot } /]$ $[role=agent \text{ by } [ne=person_description \text{ anarchist}]]$ $[ne=person \text{ Leon Czolgosz}]$ $[role=location \text{ at the } [ne=event \text{ Pan-American Exposition}]]$ in $[ne=us_city \text{ Buffalo}]$, $[ne=us_state \text{ N.Y.}]$ $[ne=person \text{ McKinley}]$ died $[ne=date \text{ eight days later}]$.

Figure 1.2: An example which shows how semantic role patterns can help better rank a sentence containing the correct answer.

Chapter 2

History of Computational Semantics

The objective of this chapter is three fold: i) To delve into the the beginnings of investigations into the nature and necessity of semantic representations of language, that would help explicate its behavior, and the various theories that were developed in the process, ii) To recount the historical approaches to interpretation of language by computers, and iii) Of the many bifurcations in recent years, those that specifically deal with automatically identifying semantic roles in text, with an end to facilitate text understanding.

2.1 The Semantics View

The farthest we might have to trace back in history to suffice the discussion pertaining to this thesis, is to the seminal paper by Chomsky – *Syntactic Structures*, which appeared in 1957, and introduced the concept of a transformational phrase structure grammar to provide an operational definition for the combinatorial formations of meaningful natural language sentences by humans. This was to be followed by the first published work treating semantics within the generative grammar paradigm, by Katz and Fodor (1963). They found that Chomsky’s (1957) transformational grammar was not a complete description of language, as it did not account for *meaning*. In their paper *The Structure of a Semantic Theory* (1963), they tried to put forward, what they thought were the properties that a semantic theory should possess. They felt that such

a theory should be able to:

- (1) Explain sentences having ambiguous meanings. For example, it should account for the fact that the word *bill* in the sentence *The bill is large.* is ambiguous in the sense that it could represent money, or the beak of a bird.
- (2) Resolve the ambiguities looking at the words in context, for example, if the same sentence is extended to form *The bill is large, but need not be paid,* then the theory should be able to disambiguate the monetary meaning of *bill*.
- (3) Identify meaningless, but syntactically well-formed sentences, such as the famous example by Chomsky – *Colorless green ideas sleep furiously,* and
- (4) Identify that syntactically, or rather transformationally, unrelated paraphrases of a concept having the same semantic content.

To account for these aspects, they presented a *interpretive* semantic theory. Their theory has two postulates:

- (1) Every lexical unit in the language, as small as a word, or combination thereof forming a larger constituent, has its semantics represented by *semantic markers* and *distinguishers* so as to distinguish it from other constituents, and
- (2) There exist a set of *projection rules* which are used to compositionally form the semantic interpretation of a sentence, in a fashion similar to its syntactic structure, to get to it's underlying meaning.

Their best known example is of the word *bachelor* which is as follows:

bachelor, [+N, ...], (Human), (Male), [who has never married]
 (Human), (Male), [young knight serving ...]
 (Human), [who has the first or lowest academic degree]
 (Animal), (Male), [young fur seal ...]

The words enclosed in parentheses are the *semantic markers* and the descriptions enclosed in square brackets are the *distinguishers*.

Subsequently, Katz and Postal (1964) argued that the input to these so called *projection rules* should be the DEEP syntactic structure, and not the SURFACE syntactic structure. DEEP syntactic structure is something that exists between the actual semantics of a sentence and its SURFACE representation, which is gotten by applying various *meaning preserving* transformation rules to this DEEP structure. In other words, two synonymous sentences should have the same DEEP structure. The DEEP structure is simultaneously subject to two sequence of rules – the *transformational rules* that convert it to the surface representation, and the *projection rules* that interpret its meaning. The fundamental idea behind this interpretation is that words, incrementally disambiguate themselves owing to incompatibility introduced by the *selection restrictions* imposed by the words, or constituents that they combine with. For example if the word *colorful* has two senses and the word *ball* has three senses, then the phrase *colorful ball* can potentially have six different senses, but if it is assumed that one of the sense of the word *colorful* is not compatible with two senses of the word *ball*, then those will automatically be excluded from the joint meaning, and only four different readings of the term *colorful ball* will be retained, instead of six. Further, the set of readings for the sentence, as a whole, after such intermediate terms are combined together, will finally contain one or many meaning elements. If it happens that the sentence is unambiguous, then the final set will comprise only one meaning element, which will be the meaning of that sentence. It could also happen that the final set has multiple readings, which means that the sentence is inherently ambiguous, and some more context might be required to disambiguate it completely. The Katz-Postal hypothesis was later incorporated by Chomsky in his *Aspects of the Theory of Syntax* (1965) which came to be known as the STANDARD THEORY.

In the meanwhile, another school of thought was lead by McCawley (1968), who questioned the existence of the DEEP syntactic structure, and claimed that syntax and semantics cannot be separated from each other. He argued that the surface level representations are formed by applying transformations directly to the core semantic representation. McCawley supported this with two arguments:

- (1) There exist phenomena that demand that an abolition of the independence of syntax from semantics, as explaining these in terms of traditional theories tends to hamper significant generalizations, and
- (2) Working under the confines of the Katz-Postal hypothesis, linguists are forced to make ever more abstract associations to the DEEP structure.

Consider the following two sentences:

- (a) The door opened.
- (b) Charlie opened the door.

In (a) the grammatical subject of *open* is *the door* and in (b) it is its grammatical object. However, it plays the same semantic role in both cases. Traditional grammatical relations fall short of explaining this phenomenon. If the Katz-Postal hypothesis is to be accepted, then there needs to be an *abstract* DEEP structure that identifies *the door* with the same semantic function. One solution that was proposed by Lakeoff (1971) and later adopted by generative semanticists was to break part of the semantic readings and express those in terms of a higher PRO-verb, that gets eventually deleted. For example, the sentence (b) above can be re-written as

- (c) Charlie caused the door to open.

Thus, maintaining the same functional relation of *the door* to *open*. However, not all sentences can be represented as this (cf. page 28, Jackendoff, 1972). The contention

was that once the structures are allowed to contain semantic elements at their leaf nodes, the DEEP syntactic structures can themselves serve as the semantic representations, and the *interpretive* semantic component can be dispensed with – thus the name *Generative Semantics*.

The idea of completely dissolving the DEEP syntactic structures was resisted by Chomsky. His argument was that there are sentences having the same semantic representations that exhibit significant syntactic differences that are not naturally captured with a difference in the transformational component, but demand the presence of an intermediate DEEP structure which is different from the semantic representation, and that the grammar should contain an *interpretive* semantic component. “The first and the most detailed argument of this kind is contained in *Remarks on Nominalizations* (1970), where Chomsky disputed the common claim that a ‘derived nominal’ such as (a) should be derived transformationally from the sentential deep structure (b)

(a) John’s eagerness to please.

(b) John is eager to please.

Despite the similarity of meaning and, to some extent, of structure between these two expressions, Chomsky argued that they are not transformationally related.” (Fodor, J. D., 1963)

In *Deep Structure, Surface Structure, and Semantic Interpretation* (1970), Chomsky gives another argument against McCawley’s proposal. Consider the following three sentences:

(a) John’s uncle.

(b) The person who is the brother of John’s mother or father or the husband of the sister of John’s mother or father

(c) The person who is the son of one of John’s grandparents or the husband of a daughter of one of John’s grandparents, but is not his

father.

Now, consider the following snippet (d) appended to the beginning of each of the above three sentences to form sentences (e), (f) and (g)

(d) Bill realized that the bank robber was --

Obviously, although (a), (b) and (c) can be considered to be paraphrases of each other, sentences (e), (f) and (g) would not be so. Now, let's consider (a), (b) and (c) in the light of the standard theory. Each of them would be derived from different DEEP structures which map on to the same semantic representation. In order to assign different meanings to (e), (f) and (g), it is important to define *realize* such that the meaning of *Bill realized that p* depends not only on the semantic structure of *p*, but also on the deep structure of *p*. In case of the standard theory, there does not arise any contradiction for this formulation. Within the framework of a semantically-based theory, however, since there is no DEEP structure, there is only a same semantic representation that represents sentences (a), (b) and (c) and it is impossible to fulfill all the following conditions:

- (1) (a), (b) and (c) have the same representation
- (2) (e), (f) and (g) have different representations
- (3) "The representation of (d) is independent of which expression appears in the context of (d) at the level of structure at which these expressions (a), (b) and (c) differ." (Chomsky, *Deep Structure, Surface Structure, and Semantic Interpretation*, 1970; reprinted in 1972, pg. 86–87)

Therefore, the semantic theory alternative collapses.

In the meanwhile, Jackendoff (1972) proposed that a semantic theory should contain the following four components:

- (1) Functional Structure (or, predicate-argument structure)
- (2) Table of coreference, which contains pairs of referring items in the sentence.
- (3) Modality, which specifies the relative scopes of elements in the sentence, and
- (4) Focus and presupposition, which specifies “what information is intended to be new and what information is intended to be old” (Jackendoff, 1972)

Chomsky later incorporated Jackendoff’s components into the standard theory, and stated a new version of the standard theory – the EXTENDED STANDARD THEORY (EST)

The schematic in Figure 2.1 illustrates the structure of the Aspects Theory, or the STANDARD THEORY which incorporates the Katz-Postal hypothesis.

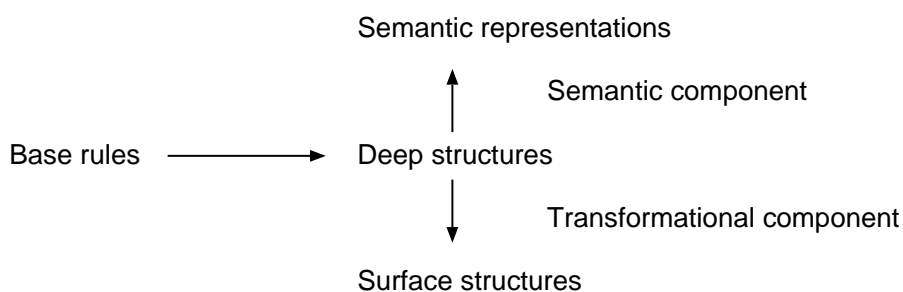


Figure 2.1: Standard Theory.

The schematic in Figure 2.2 illustrates the structure of the EXTENDED STANDARD THEORY.

Subsequently, there were several refinements to the EST, which resulted in the REVISED EXTENDED STANDARD THEORY (REST), followed by the GOVERNMENT and BINDING THEORY (GB). We need not go into the details of these theories for purposes of this thesis.

Traditional linguistics considers case as mostly concerned with the morphology of nouns. Fillmore, in his *Case for Case* (1968) states that this view is quite narrow-

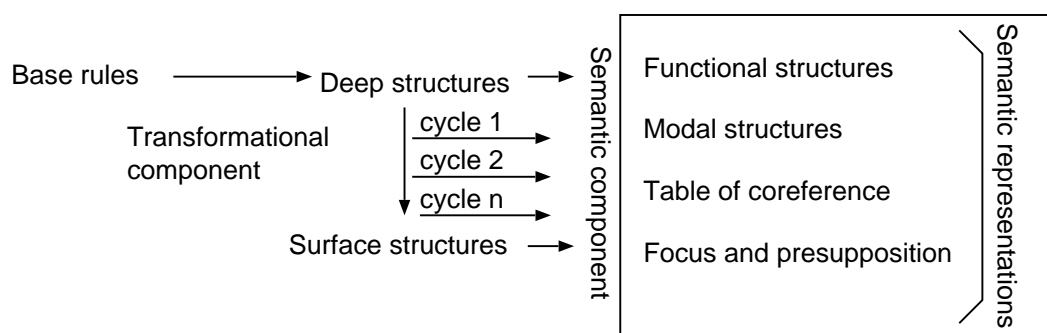


Figure 2.2: Extended Standard Theory.

mindful, and that lexical items such as prepositions, syntactic structure, etc. exhibit the a similar relationship with nouns or noun phrases and verbs in a sentence. He rejects the traditional categories of *subject* and *object* as being the semantic primitives, and gives them a status closer to the surface syntactic phenomenon, rather than accepting them as being part of the DEEP structure. He rather considers the case functions to be part of the DEEP structure. He justifies his position using the following three examples:

- (a) John opened the door with a key.
- (b) The key opened the door.
- (c) The door opened.

In these examples the subject position is filled by three different participants in the same action of opening the door. Once by *John*, once by *The key*, and once by *The door*.

He proposed a CASE GRAMMAR to account for this anomaly. The general assumptions of the case grammar are:

- In a language, simple sentences contain a proposition and a modality component, which applies to the entire sentence.
- The proposition consists of a verb and its participants. Each case appears in the surface form as a noun phrase.

- Each verb instantiates a set of finite cases.
- For each proposition, a particular case appears only once.
- The set of cases that a verb accepts is called its case frame. For example, the verb *open* might take a case frame (AGENT, OBJECT, INSTRUMENT).

He enumerated the following primary cases. He envisioned that more would be necessary to account for different semantic phenomenon, but these are the primary ones:

AGENTIVE – Usually the animate entity participating in an event.

INSTRUMENTAL – Usually an inanimate entity that is involved in fulfilling the event.

DATIVE – The animate being being affected as a result of the event

FACTITIVE – The object or being resulting from the instantiation of the event

OBJECTIVE – “The semantically most neutral case, the case of anything representable by a noun whose role in the action or state identified by the verb is identified by the semantic interpretation of the verb itself; conceivably the concept should be limited to the things which are affected by the action or state identified by the verb. The term is not to be confused with the notion of direct object, nor with the name of the surface case synonymous with accusative” (Case for Case, Fillmore, 1968, pages 24-25)

LOCATIVE – This includes all cases relating to locations, but nothing that implies directionality.

Around the same time a system was developed by Gruber in his dissertation and other work (Gruber, 1965; 1967) which, superficially looks exactly like Fillmore’s case roles, but differs from it in some significant ways. According to Gruber, in each

sentence, there is a noun phrase that acts as a *Theme*. For example, in the following sentences with motion verbs (examples are from Jackendoff (1972)) the object that is set in motion is regarded as the *Theme*

- (a) The rock moved away.
- (b) John rolled the rock from the dump to the house.
- (c) Bill forced the rock into the hole.
- (d) Harry gave the book away.

Here, *the rock* and *the book* are *Themes*. Note that the *Theme* can be either grammatical *subject* or *object*. In addition to *Theme*, Gruber also discusses some other roles like *Agent*, *Source*, *Goal*, *Location*, etc. The *Agent* is identified by the constituent that has a volitional function for the action mentioned in the sentence. Only animate NPs can serve as *Agents*. As per Gruber's analysis, if we replace *The rock* with *John* in (a) above, then *John* acts as both the *Agent* as well as the *Theme*. In this methodology, imperatives are permissible only for *Agent* subjects. There are several other analysis that Gruber goes in detail in his dissertation.

Jackendoff gives two reasons why he thinks that Gruber's theory of THEMATIC ROLES is preferred to Fillmore's CASE GRAMMAR.

First, it provides a way of unifying various uses of the same morphological verb. One does not, for example, have to say that *keep* in *Herman kept the book on the shelf* and *Herman kept the book* are different verbs; rather one can say that *keep* is a single verb, indifferent with respect to positional and possessional location. Thus Gruber's system is capable of expressing not only the semantic data, but some important generalizations in the lexicon. A second reason to prefer Gruber's system of thematic relations to other possible systems [...] It turns out that some very crucial generalizations about the distribution of reflexives, the possibility of performing the passive, and the position of antecedents for deleted complement subjects can be stated quite naturally in terms of thematic relations. These generalizations have no a priori connection with thematic relations, and in fact radically different solutions, such as Postal's Crossover Condition and Rosenbaum's Distance Principle,

have been proposed in the literature. [...] The fact that they are of crucial use in describing independent aspects of the language is a strong indication of their validity.

2.2 The Computational View

While, linguists and philosophers were trying to define what the term *semantics* meant, and were trying to crystallize its position in the architecture of language, there were computer scientists who were curious about making the computer understand natural language, or rather program the computer in such a way that it could be useful for some specific tasks. Whether or not it really understood the cognitive side of language was irrelevant. In other words, their motivation was not to solve the philosophical question of *what does semantics entail?*, but rather to try to make the computer solve tasks that have roots in language – with or without using any affiliation to a certain linguistic theory, but maybe utilizing some aspects of linguistic knowledge that would help encode the task at hand in a computer.

2.2.1 BASEBALL

This is a program originally conceived by Frick, Selfridge (Simmons, 1965) and implemented by Green et al. (1961, 1963). It stored a database of baseball games as shown in figure 2.3 and could answer questions like *Who did the Red Sox play on July 7?* The program first performs a syntactic analysis of the question and determined the noun phrases and prepositional phrases, and the identities of the subjects and objects. Later in the semantic analysis phase, it generates a specification list that is sort of a template with some fields filled in and some blank. The blank fields usually are the once that are filled with the answer. Then it runs it routine to fill the blank fields using a simple matching procedure. In his 1965 survey Robert F. Simmons says “Within the limitations of its data and its syntactic capability, Baseball is the most sophisticated and successful of the first generation of experiments with question-answering machines”

MONTH	PLACE	DAY	GAME	WINNER/SCORE	LOSER/SCORE
July	Cleveland	6	95	White Sox/2	Indians/0
July	Boston	7	96	Red Sox/5	Yankees/3
July	Detroit	7	97	Tigers/10	Athletics/2

Figure 2.3: Example database in BASEBALL

Some other similar programs that were developed around the same time were **SAD-SAM**, **STUDENT**, etc. **SAD-SAM** (Sentence Appraiser and Diagrammer and Semantic Analyzing Machine) was a program written by Lindsay (1963) as part of his dissertation at Carnegie Institute of Technology. It accepted a low vocabulary sentences of English which described family relationships and built an interlocking structure which allowed a significant level of inference. “[...] Lindsay’s primary interest was in machine comprehension of English and he attempted to show that an important component of understanding lay in building large coordinated data structures from the text which was read.” (Simmons, 1965). **STUDENT** (Bobrow, 1964, 1968) was a program written by Daniel Bobrow. It could solve high-school level algebra problems by converting the English input to a formal representation.

2.2.2 ELIZA

Joseph Weizenbaum wrote a program called ELIZA (Weizenbaum, 1966) (inspired by the character Eliza in Pygmalion). This is probably the first program that tried to emulate human language understanding by a computer. It was a small program with some 200 lines of code, and used regular expression patterns to process input and provide a coherent-seeming dialog as if the user is talking with a psychiatrist. At the heart of the program was a script that identified keywords in the conversation and ranked them using a predefined ranking scheme. The highest ranking keyword in an input sentence

invoked a transformation rule that generated an seemingly *intelligent* response. The fact remained that the system made no attempt at real comprehension. When no keywords are identified, the system defaults to some standard response. Figure 2.4 shows a sample conversation with Eliza:

```
USER: Men are all alike
ELIZA: In what way
USER: They're always bugging us about something or other
ELIZA: Can you think of a specific example
USER: Well, my boyfriend made me come here
ELIZA: Your boyfriend made you come here
USER: He says I'm depressed much of the time
ELIZA: I am sorry to hear you are depressed
USER: It's true. I am unhappy
ELIZA: Do you think coming here will, help you not to be unhappy
USER: I need some help, that much seems certain
ELIZA: What would it mean to you if you got some help
USER: Perhaps i could learn to get along with my mother
ELIZA: Tell me more about your family
USER: My mother takes care of me
ELIZA: Who else in yOur family takes care of you
USER: My father
ELIZA: Your father
USER: You are like my father in some ways
ELIZA: What resemblance do you see
USER: You are not very aggressive but I think you don't want me to
notice that
ELIZA: What makes you think I am not very aggressive
USER: You don't argue with me
ELIZA: Why do you think i don't argue with you
USER: You are afraid of me
ELIZA: Does it please you to believe i am afraid of you
USER: My father is afraid of everybody
ELIZA: What else comes to mind when you think of your father
USER: Bullies
ELIZA: Does that have anything to do with the fact that your boyfriend
made you come here
```

Figure 2.4: Example conversation with ELIZA.

2.2.3 SHRDLU

“The years around 1970 proved to be noteworthy for a number of reasons. I describe briefly several well known and influential programs that appeared around 1970, and that pushed the notion of semantic information processing to its ultimate limits” (Waltz, 1982). The first one of those is Winograd’s SHRDLU (Winograd, 1971, 1972). The primary assumption of Winograd was that sentences could be converted in to programs and these programs could be used for various tasks, for example, moving blocks of various geometries, placed on a table. It used a heuristic search which generated a list of possible understandings of a sentence, and depending on whether a certain hypothesis made sense, it backed up to another hypothesis until it made syntactic and semantic sense. It used the microPLANNER programming language (Sussman et al., 1971) which was inspired by the PLANNER language (Hewitt, 1970). The novelty of SHRDLU compared to systems of those days was that it could handle a wide-variety of natural language sentences – interrogatives, declaratives and imperatives, and it could handle semantic phenomena like – quantification, pronoun reference, negation, etc., to a certain degree. Figure 2.5 shows a sample interaction with SHRDLU.

```
USER: Find a block which is taller than the one you are holding and  
put it into the box.  
SHRDLU: By ‘‘it,’’ I assume you mean the block which is taller than  
the one I am holding. O.K.
```

Figure 2.5: Example interaction in SHRDLU

2.2.4 LUNAR

Around the same time William Woods and his colleagues built the LUNAR system (Woods, 1977, 1973). This was a natural language front end to a database that contained

scientific data of moon rock sample analysis. Augmented Transition Networks (Woods, 1967, 1970) was used to implement the system. It consisted of heuristics similar to those in Winograd's SHRDLU. "Woods' formulation was so clean and natural that it has been used since then for most parsing and language-understanding systems" (Waltz, 1982). It introduced the notion of procedural semantics (Woods, 1967) and had a very general notion of quantification based on predicate calculus (Woods, 1978) An example question that Woods' LUNAR system could answer is "Give me all analyzes of sample containing olivine." (Waltz, 1982)

2.2.5 NLPQ

Another program that came out during that time was the work of George Heidorn and was called NLPQ Heidorn (1974). It used natural language interface to let the user set up a simulation and could run it to answer questions. An example of the simulations would be a time study of the arrival of vehicles at a gas station. The user could set up the simulation and the system would run it. Subsequently, the user could ask question such as *How frequently do the vehicles arrive at the gas station?* etc.

2.2.6 MARGIE

Schank (1972) introduced the theory of *Conceptual Dependency*, which stated that the underlying nature of language is conceptual. He theorized that there are the following set of cases between actions (A) and nominals (N). A case is represented by the shape of an arrow and its label. Following are the conceptual cases that he envisioned – ACTOR, OBJECTIVE, RECIPIENT, DIRECTIVE and INSTRUMENT. Schank's conceptual case can, in some ways, be related to Fillmore's cases, but there are some important distinctions as we shall see later. They are diagrammatically represented as shown in Figure 2.6

The second component is a set of some 16 conceptual-dependency primitives as

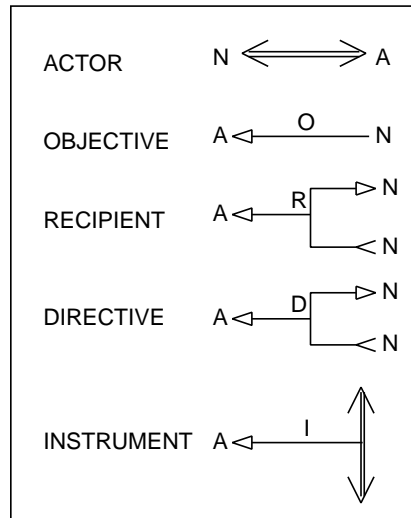


Figure 2.6: Schank's conceptual cases.

shown in Table 2.1 which are used to build the dependency structures.

Schank hypothesized certain properties of this conceptual-dependency representation:

- (1) It would not change across languages,
- (2) Sentences with the same *deep structure* would be represented with the same structure, and
- (3) It would provide an intermediate representation between a surface structure and a logical formula, thus simplifying potential proofs.

He built a program called MARGIE (Meaning Analysis, Response Generation and Inference on English) (Schank et al., 1973), which could accept English sentences and answer questions about them, generate paraphrases and perform inference on them. Figure 2.7 shows a conceptual-dependency structure representing the sentence *The big boy gives apples to the pig.*

Primitive	Description
ATRANS	- Transfer of an abstract relationship. e.g. give.
PTRANS	- Transfer of the physical location of an object. e.g. go.
PROPEL	- Application of a physical force to an object. e.g. push.
MTRANS	- Transfer of mental information. e.g. tell.
MBUILD	- Construct new information from old. e.g. decide.
SPEAK	- Utter a sound. e.g. say.
ATTEND	- Focus a sense on a stimulus. e.g. listen, watch.
MOVE	- Movement of a body part by owner. e.g. punch, kick.
GRASP	- Actor grasping an object. e.g. clutch.
INGEST	- Actor ingesting an object. e.g. eat.
EXPEL	- Actor getting rid of an object from body.

Table 2.1: Conceptual-dependency primitives.

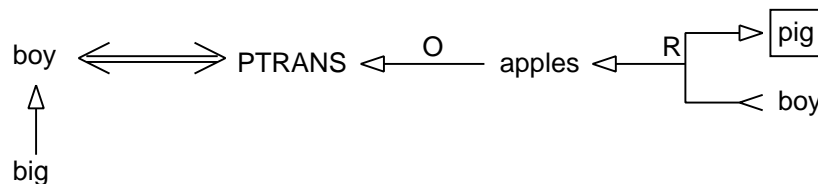


Figure 2.7: Conceptual-dependency representation of "The big boy gives apples to the pig."

Schank's concept of cases is different from the case frames proposed by Fillmore (1968) in the following ways, as Samlowski (1976) indicates:

- (1) Fillmore's case frames are for surface verbs [...] Schank, however, specifies case frames for each of his twelve basic verbs which he calls *primitive acts* [...]
- (2) Fillmore's cases, in frames, can [...] be either obligatory or optional. Schank insists that for any primitive act, its associated cases are obligatory.
- (3) As we would expect from 2, since all the cases for a given act must be filled in a conceptualization, this will mean the insertion of participants not necessarily mentioned in the surface sentence. Fillmore originally restricted his representations to mentioned participants, but more recently has accommodated the presence of null instantiations in his theory.

Techniques based on conceptual dependency representation were quite successful in limited domain applications, but had a problem scaling up to open-domain natural language complexity and ambiguity. One recent successor of this paradigm for language understanding is the *Core Language Engine* of Alshawi (1992) where the representation is based on a *Quasi-Logical Form* that includes quantifier scoping, coreference resolution and long distance dependencies.

Once it came to be a widely widely accepted fact that natural language understanding is an AI-complete problem, the focus of researchers shifted from the search for a complete, deep semantic representation, to a more rule-based one utilizing linguistic properties or frequent patterns found in a corpora. This new era also saw a series of Message Understanding Conferences (MUC) that led to systems like FASTUS (Hobbs et al., 1997) – a cascaded finite state system that analyzes successive chunks of text using hand-written rules. ALEMBIC (Aberdeen et al., 1995), PROTEUS (Grishman et al., 1992; Yangarber and Grishman, 1998), KERNEL (Palmer et al., 1993) were some other rule-based systems that could perform the tasks of named entity identification, coreference resolution etc. Further improvement in this methodology was brought by the AutoSlog (Riloff, 1993) system that could automatically learn semantic dictionaries for a domain, and a latter version – AutoSlog-TS (Riloff, 1996; Riloff and Jones, 1999), which used unsupervised methods to generate patterns.

While a group of researchers were building domain specific dictionaries and incorporating methods to automatically learn those, some others were generating large corpora which could facilitate the formulation of the natural language understanding task as a pattern-recognition problem and use supervised learning algorithms to solve it. This marked another era in the development of natural language understanding systems. It experienced the generation of two major linguistic corpora – the British National Corpus (BNC) which contains about a 100 million words of written (90%) as well as spoken (10%) British English, segmented into sentences and tagged with part of

speech information using the C5 tagset. The Penn Treebank (Marcus et al., 1994b) – a million word corpus, annotated with detailed syntactic information. Ratnaparkhi (1996) part of speech tagger, and Collins (1997) and Charniak (2000) syntactic parsers are some manifestations of the application of statistical and machine learning techniques to the Treebank. It is envisioned that one day it would be possible to combine the solutions to these sub-problems into robust, scalable, natural language understanding systems.

2.3 Early Semantic Role Labeling Systems

Early semantic role labeling programs can be traced back to Warren and Friedman (1982)'s semantic equivalence parsing for non-context-free parsing of Montague grammar, which collapses syntax and semantics in an essentially domain-specific semantic grammar. Hirst (1983)'s Absity semantic parser – based on a variation of Montague-like semantics, but replaces the semantic objects, functors and truth conditions with elements of a frame language called Frail, and adds a word sense, and case slot disambiguation system to it; and Sondheimer et al. (1984)'s semantic interpreter engine that uses semantic networks – KL-ONE. Despite the fact that the detail representation of semantics provided a powerful medium for language understanding, they were quite impractical for real-world applications, therefore researchers started pursuing the idea of automatically identifying semantic argument structures, exemplified by the thematic roles or case roles, as a step towards domain independent semantic interpretation. Also, since they are most closely linked to syntactic structures (Jackendoff, 1972), their recognition could be feasible given only syntactic cues. One notable exception to this is the CYC¹ database that is attempting to capture a vast amount of real-world knowledge over the past few decades. McClelland and Kawamoto (1986) connectionist model of thematic role assignment using semantic micro-features for a small number of verbs and nouns was probably the first attempt to identify thematic roles in text. Liu and Soo

¹ <http://www.cyc.com>

(1993) present an heuristic approach to identify thematic argument structures for verbal predicates. Their heuristics are based on features extracted from the syntactic parse of a sentence, for example, the constituent type (they only consider noun phrases, adverbial phrases, adjective phrases and prepositional phrases as possible argument instantiators), animacy, grammatical function and prepositions in the prepositional phrases. They also use some heuristics like the *thematic hierarchy heuristic* following the *thematic hierarchy condition* put forward by Jackendoff (1972), a *uniqueness heuristic* which states that no two arguments in the same clause, can take the same semantic role, etc. Rosa and Francozo (1999) present a connectionist learning mechanism using hybrid knowledge-based neural network to identify thematic grids for a small set of verbs. These efforts were done before there was any comprehensive semantically annotated corpus. In addition to the syntactic structure provided to sentences in The Penn Treebank, there are also some semantic tags that are assigned to phrases. Some examples of these are subject (SUB), object (OBJ), location (LOC), etc. Blaheta and Charniak (2000) report a method to recover these semantic tags using a feature tree with features such as the part of speech, head word, phrase type, etc., in a maximum entropy framework.

2.4 Advent of Semantic Corpora

The late 90s saw the emergence of two main corpora that are semantically tagged. One is called FrameNet² (Baker et al., 1998; Fillmore and Baker, 2000) which is due to Fillmore, who has expanded his ideas to “frame semantics” – where, a given predicate invokes a “semantic frame”, thus instantiating some or all of the possible semantic roles belonging to that frame. Another such project is the PropBank³ (Palmer et al., 2005a) which has annotated the Penn Treebank with semantic information. Following is an overview of the two corpora.

² <http://www.icsi.berkeley.edu/~framenet/>

³ <http://www.cis.upenn.edu/~ace/>

FrameNet contains *frame*-specific semantic annotation of a number of predicates in English. It tags sentences extracted from the BNC corpus. The process of FrameNet annotation consists of identifying specific semantic frames and creating a set of frame-specific roles such as **SPEAKER** and **MESSAGE**. Then a subset of predicates that instantiate that semantic frame, irrespective of their grammatical category, are selected, and sufficient variety of sentences are labeled for those predicates. The labeling process entails identifying the semantic arguments of the predicate and tagging them with one of the frame-specific roles. The current release of FrameNet contains about 500 different frame types and about 700 distinct frame elements. Following example illustrates the general idea. Here, the frame **STATEMENT** is instantiated by the predicate *complain*, once as a nominal predicate and once as a verbal predicate.

1. Did [*Speaker* she] make an official [*Predicate:nominal* complaint] [*Addressee* to you] [*Topic* about the attack.]
2. [*Message* “Justice has not been done”] [*Speaker* he] [*Predicate:verbal* complained.]

The nominal predicates in FrameNet encompass a wide variety. They include ultra-nominals (Barker and Dowty, 1992; Dowty, 1991), nominals and nominalizations.

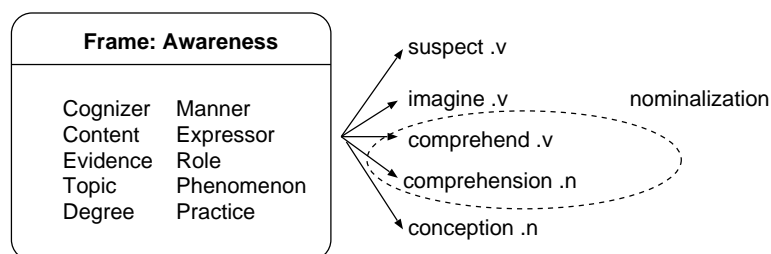


Figure 2.8: FrameNet example.

PropBank, on the other hand, only annotates arguments of verb predicates. All the non-copula verbs in the Wall Street Journal (WSJ) part of the Penn Treebank (Marcus et al., 1994a) have been labeled with their semantic arguments. Furthermore, it restricts the argument boundaries to that of one of the syntactic constituent as defined in the Penn Treebank. There are very few (1-2%) instances where the arguments are split across a predicate, in which case, multiple nodes in the correct syntax tree are marked with the same argument. It uses a linguistically neutral terminology to label the arguments. The arguments are tagged as either being *core arguments*, with labels of the type ARG N where N takes values from 0 to 5, or *adjunctive arguments*, with labels of the type ARG M - X , where X can take values such as TMP for temporal, LOC for locative, etc. Adjunctive arguments share the same meaning across all the predicates, whereas the meaning of core arguments has to be interpreted in connection with a predicate. ARG0 is the PROTO-AGENT (usually the subject of a transitive verb), ARG1 is the PROTO-PATIENT (usually its direct object of the transitive verb). Table 2.2 shows a list of core arguments for the predicate *operate*, and Table 2.3 shows the same for the predicate *author*. Note that some core arguments – ARG2, ARG3, etc. do not occur with this predicate. This is explained by the fact that not all the core arguments can be instantiated by all the senses of a predicate. A list of core arguments that can occur with a particular sense of the predicate, along with their real world meaning, is present in a file called the *frames* file. There is one such frames file associated with all senses of each predicate.

Following is an example structure extracted from PropBank corpus. The syntax tree representation along with the argument labels is shown in Figure 2.9.

Sometimes the tree can have *trace* nodes which refer to another node in the tree,

Tag	Description
Arg0	Agent, operator
Arg1	Thing operated
Arg2	Explicit patient (thing operated on)
Arg3	Explicit argument
Arg4	Explicit instrument

Table 2.2: Argument labels associated with the predicate *operate* (sense: work) in the PropBank corpus.

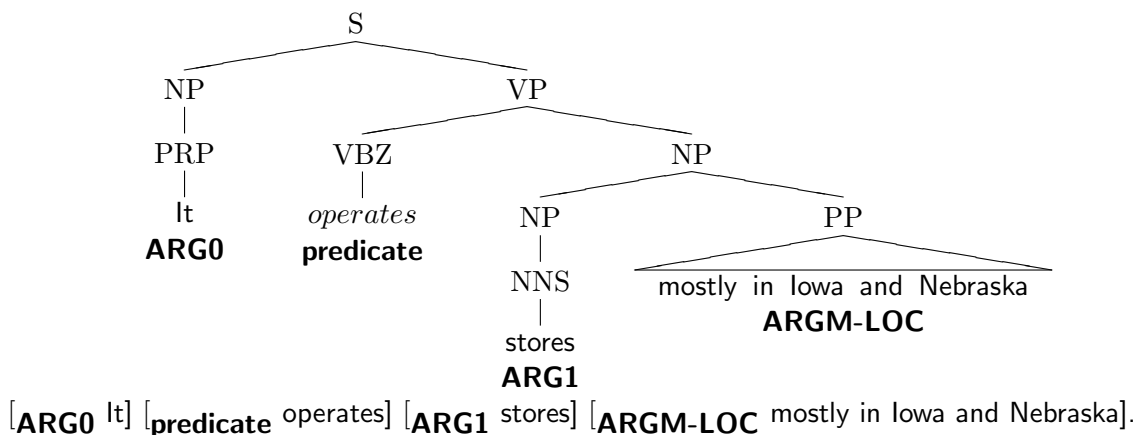


Figure 2.9: Syntax tree for a sentence illustrating the PropBank tags

but do not have any words associated with them. These can also be marked as arguments. As traces are not reproduced by a syntactic parser, we decided not to consider them in our experiments – whether or not they represent arguments of a predicate. PropBank also contains arguments that are coreferential.

2.5 Corpus-based Semantic Role Labeling

It is clear from the foregoing discussion that imparting some level of semantic structure to plain text makes it amenable to application of techniques that can emulate natural language understanding. History also indicates that any effort to encoding semantic information at a very fine grained level using a bottom-up cumulative encoding of domain-dependent semantic information with an end to achieving domain-independence

Tag	Description
Arg0	Author, agent
Arg1	Text authored

Table 2.3: Argument labels associated with the predicate *author* (sense: to write or construct) in the PropBank corpus.

seems infeasible. A prominent example being CYC, which, even after 20 years of continuous tagging at encoding common-sense knowledge, has not experienced wide-scale acceptance by the research community. Also, there is no evidence of any rule-based technique developed over the years that covers all the domain independent semantic nuances at even a shallow level of detail in any single language. What this tells us that such attempts quickly fail to scale-up – both in terms of space and time, and a top-down approach to capturing semantic details at a high-level domain independently might indicate the right direction. This hypothesis is supported by the fact that prominent researchers in the field have taken a stance towards creating manually tagging corpora that encode shallow semantic information, as described in the preceding section. At this point there appears to be a consensus within the community that, at least for the near future, an approach to this problem might lie in generating domain-independent, semantically encoded corpora, and devising machine learning algorithms to create taggers that would reproduce such encoding on unseen/untagged text. Furthermore, this could potentially reduce the lead time for engineering domain-specific systems by quickly generating skeleton schema for those domains. These attempts mark the current generation of what has come to be popularly known today as “semantic role labeling” systems. In this chapter we will look at the details of some of the early systems that used supervised machine learning algorithms to learn semantic roles from tagged corpora, starting with the system described by Gildea and Jurafsky (2002), which marked the beginning of this new era.

Table 2.4: List of adjunctive arguments in PropBank – ARGMS

Tag	Description	Examples
ARGM-LOC	Locative	<i>the museum, in Westborough, Mass.</i>
ARGM-TMP	Temporal	<i>now, by next summer</i>
ARGM-MNR	Manner	<i>heavily, clearly, at a rapid rate</i>
ARGM-DIR	Direction	<i>to market, to Bangkok</i>
ARGM-CAU	Cause	<i>In response to the ruling</i>
ARGM-DIS	Discourse	<i>for example, in part, Similarly</i>
ARGM-EXT	Extent	<i>at \$38.375, 50 points</i>
ARGM-PRP	Purpose	<i>to pay for the plant</i>
ARGM-NEG	Negation	<i>not, n't</i>
ARGM-MOD	Modal	<i>can, might, should, will</i>
ARGM-REC	Reciprocals	<i>each other</i>
ARGM-PRD	Secondary Predication	<i>to become a teacher</i>
ARGM	Bare ARGM	<i>with a police escort</i>
ARGM-ADV	ADVERBIALS	<i>(none of the above)</i>

2.5.1 Problem Description

The process of semantic role labeling can be defined as identifying a set of word sequences each of which represents a semantic argument of a given predicate. For example in the sentence below,

(1) I 'm inspired by the mood of the people

For the predicate “inspired,” the word “I” represents the ARG1 and the sequence of words “the mood of the people” represents ARG0

2.6 The First Cut

Gildea and Jurafsky (2002) defined the problem of semantic role labeling as a classification of nodes in a syntax tree, with the assumption that there is a one-to-one mapping between arguments of a predicate and nodes in the syntax tree. They introduced three tasks that could be used to evaluate the system. They are as follows:

Argument Identification – This is the task of identifying all and only the parse

constituents that represent valid semantic arguments of a predicate.

Argument Classification – Given constituents known to represent arguments of a predicate, assigning the appropriate argument labels to them.

Argument Identification and Classification – Combination of the above two tasks, where the constituents that represent arguments of a predicate are identified, and the appropriate argument label assigned to them.

A phrase structure parse of the above example is shown in example below.

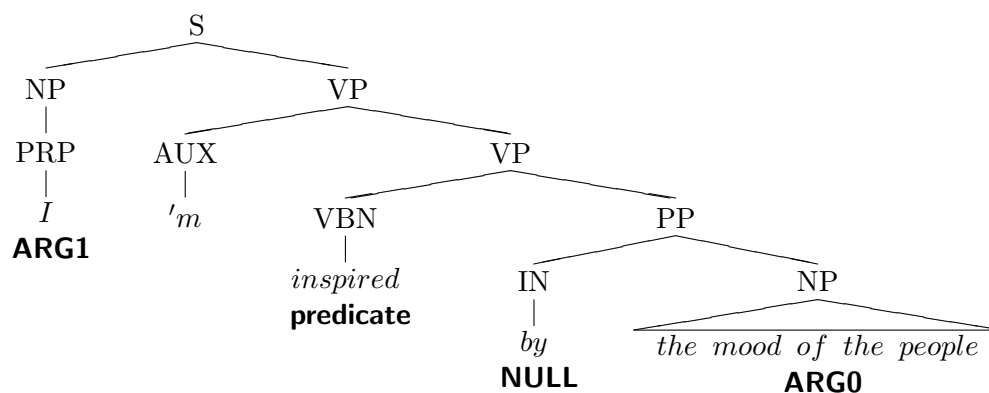


Figure 2.10: A sample sentence from the PropBank corpus

Once the sentence has been parsed using a parser, each node in the parse tree can be classified as either one that represents a semantic argument (i.e., a NON-NULL node) or one that does not represent any semantic argument (i.e., a NULL node). The NON-NULL nodes can then be further classified into the set of argument labels.

For example, in the tree of Figure 2.10, the PP that encompasses “*by the mood of the people*” is a NULL node because it does not correspond to a semantic argument. The node NP that encompasses “*the mood of the people*” is a NON-NULL node, since it does correspond to a semantic argument – ARG0.

They used the following features, some of which were extracted from the parse tree of the sentence.

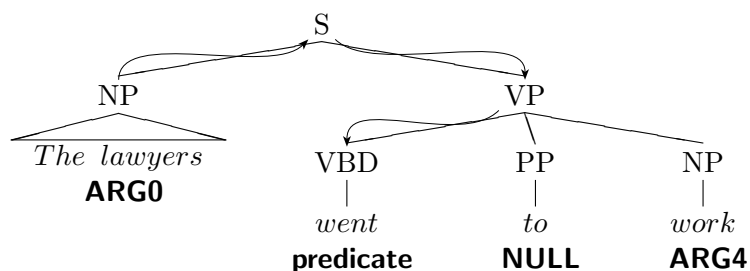


Figure 2.11: Illustration of path $\text{NP}\uparrow\text{S}\downarrow\text{VP}\downarrow\text{VBD}$

Path – The syntactic path through the parse tree from the parse constituent to the predicate being classified. For example, in Figure 2.11, the path from ARG0 – “*The lawyers*” to the predicate “*went*”, is represented with the string $\text{NP}\uparrow\text{S}\downarrow\text{VP}\downarrow\text{VBD}$. \uparrow and \downarrow represent upward and downward movement in the tree respectively.

Predicate – The predicate lemma is used as a feature.

Phrase Type – This is the syntactic category (NP, PP, S, etc.) of the constituent.

Position – This is a binary feature identifying whether the phrase is before or after the predicate.

Voice – Whether the predicate is realized as an active or passive construction. A set of hand-written `tgrep` expressions on the syntax tree are used to identify the passive voiced predicates.

Head Word – The syntactic head of the phrase. This is calculated using a head word table described by Magerman (1994) and modified by Collins (1999, Appendix. A). A case-folding operation is performed on this feature.

Sub-categorization – This is the phrase structure rule expanding the predicate’s parent node in the parse tree. For example, in Figure 2.11, the sub-

categorization for the predicate “*went*” is $VP \rightarrow VBD-PP-NP$.

They found two of the features – Head Word and Path – to be the most discriminative for argument identification. In the first step, the system calculates maximum likelihood probabilities that the constituent is an argument, based on these two features – $P(is_argument|Path, Predicate)$ and $P(is_argument|Head, Predicate)$, and interpolates them to generate the probability that the constituent under consideration represents an argument.

In the second step, it assigns each constituent that has a non-zero probability of being an argument, a normalized probability that is calculated by interpolating distributions conditioned on various sets of features, and selects the most probable argument sequence. Some of the distributions that they used are shown in the following table.

Distributions
$P(argument Predicate)$
$P(argument Phrase Type, Predicate)$
$P(argument Phrase Type, Position, Voice)$
$P(argument Phrase Type, Position, Voice, Predicate)$
$P(argument Phrase Type, Path, Predicate)$
$P(argument Phrase Type, Path, Predicate, Sub-categorization)$
$P(argument Head Word)$
$P(argument Head Word, Predicate)$
$P(argument Head Word, Phrase Type, Predicate)$

Table 2.5: Distributions used for semantic argument classification, calculated from the features extracted from a Charniak parse.

They report results on the FrameNet corpus.

2.7 The First Wave

The next couple of years saw the emergence of a few system modifications which essentially used the same formulation, but tried adding new features or using new algorithms to improve the task of semantic role labeling. We will look at a first few of those systems briefly in the following sub-sections.

2.7.1 The Gildea and Palmer (G&P) System

During this time a sufficient amount of PropBank data became available for experiments and Gildea and Palmer (2002) reported results on this set using essentially the same system and features used by Gildea and Jurafsky (2002). They used the December 2001 release of PropBank. One advantage of PropBank over FrameNet, as mentioned earlier, was the availability of perfect parses. Therefore, they could report semantic role labeling accuracies using correct syntactic information.

2.7.2 The Surdeanu *et al.* System

Following that, Surdeanu et al. (2003) reported results for a system that used the same features as Gildea and Jurafsky (2002) (Surdeanu System I), however, they replaced the learning algorithm with a decision tree classifier – C5 (Quinlan, 1986, 2003). They then reported performance gains by adding some new features (Surdeanu System II). The built-in boosting capabilities of this classifier gave a slight improvement on their performance. For their experiments, they used the July 2002 release of PropBank.

The additional features that they used were:

Content Word – It was observed that the head word feature of some constituents like PP and SBAR is not very informative and so, they defined a set of heuristics for some constituent types, where, instead of using the usual head word finding rules, a different set of rules were used to identify a so called “content” word. This was used as an additional feature. The rules that they used are shown in Figure 2.12

Part of Speech of the Content Word

Named Entity class of the Content Word – Certain roles like ARGM-TMP and ARGM-LOC tend to contain TIME or PLACE named entities. This information was added as a set of binary valued features.

- H1: if phrase type is PP then select the right-most child
 Example: phrase = "in Texas", content word = "Texas"
- H2: if phrase type is SBAR then select the left-most sentence (S*) clause
 Example: phrase = "that occurred yesterday", content word = "occurred"
- H3: if phrase type is VP then
 if there is a VP child then
 select the left-most VP child
 else
 select the head word
 Example: phrase = "had placed", content word = "placed"
- H4: if phrase type is ADVP then select the right-most child, not IN or TO
 Example: phrase = "more than", content word = "more"
- H5: if phrase type is ADJP then select the right-most adjective, verb, noun or ADJP
 Example: phrase = "61 years old", content word = "61"
- H6: for all other phrase types select the head word
 Example: phrase = "red house", content word = "red"

Figure 2.12: List of content word heuristics.

Boolean Named Entity Flags – They also added named entity information as a feature. They used seven named entities, viz., PERSON, PLACE, TIME, DATE, MONEY, PERCENT, as a binary feature, where all the entities that are contained in the constituent are marked true.

Phrasal verb collocations – This feature comprises frequency statistics related to the verb and the immediately following a preposition.

2.7.3 The Gildea and Hockenmaier (G&H) System

This system has the same architecture as the Gildea and Jurafsky (2002) and Gildea and Palmer (2002) systems, but uses features extracted from a CCG grammar instead of a phrase structure grammar. Since nodes in a CCG tree poorly align with the semantic arguments of a predicate, this system is designed for identifying words (instead of constituents) that represent the semantic arguments, and the respective argument that they represent. The features used are:

Path – The equivalent of the phrase structure path is a string that is the concatenation of the category the word belongs to, the slot that it fills, and

the direction of dependency between the word and the predicate. When the category information is unavailable, the path through the binary tree from the constituent to the predicate is used.

Phrase type – This is the maximal projection of the PropBank argument’s head word in the CCG parse tree.

Predicate

Voice

Head Word

Gildea and Hockenmaier (2003) report on both the core arguments and the adjunctive arguments on the November 2002 release of the PropBank. This will be referred to as “G&H System I”.

2.7.4 The Chen and Rambow (C&R) System

Chen and Rambow (2003) also report results using decision tree classifier C4.5 (Quinlan, 1986). They report results using two different sets of features: i) Surface syntactic features much like the Gildea and Palmer (2002) system, ii) Additional features that result from the extraction of a Tree Adjoining Grammar (TAG) from the Penn Treebank. They chose a Tree Adjoining Grammar because of its ability to address long distance dependencies in text. The additional features they introduced are:

Supertag Path – This is the same as the path feature seen earlier, except that in this case it is derived from a TAG rather than from a PSG.

Supertag – This can be the tree-frame corresponding to the predicate or the argument.

Surface syntactic role – This is the surface syntactic role of the argument.

Surface sub-categorization – This is the surface sub-categorization frame.

Deep syntactic role – This is the deep syntactic role of an argument.

Deep sub-categorization – This is the deep syntactic sub-categorization frame.

Semantic sub-categorization – This is the semantic sub-categorization frame.

Comparison of results between the system based on surface syntactic features (C&R System I), and the one that uses deep syntactic features (C&R System II) is performed. Unlike all the other systems discussed here, they ALSO(or ONLY?) report results on core arguments (ARG0-5).

2.7.5 The Fleischman *et al.* System

Fleischman et al. (2003) report results on the FrameNet corpus using a Maximum Entropy framework. In addition to the Gildea and Jurafsky (2002) features they use the following features in their system.

Logical Function – This is a feature that takes three values – external argument, object argument and other and is computed using some heuristics on the syntax tree.

Order of Frame Elements – This feature represents the position of a frame element relative to other frame elements in a sentence.

Syntactic Pattern – This is also generated using heuristics on the phrase type and the logical function of the constituent.

Previous Role

Chapter 3

Automatic Statistical SEmantic Role Tagger – ASSERT

This chapter describes the design and implementation of ASSERT – an Automatic Statistical SEmantic Role Tagger.

3.1 ASSERT Baseline

ASSERT extends the paradigm of Gildea and Jurafsky (2002). It treats the problem as a supervised classification task. The sentence under consideration is first syntactically parsed using the Charniak parser (Charniak, 2000). A set of syntactic features are extracted for each of the constituents in this syntactic parse with respect to the predicate. The associated semantic argument labels constitute the class represented by this feature vector. A Support Vector Machine (SVM) classifier is then trained on these data.

The baseline system uses the exact same set of features introduced by Gildea and Jurafsky (2002): predicate, path, phrase type, position, voice, head word, and verb sub-categorization. The first diversion from their algorithm was to replace the the statistical classifier with a Support Vector Machine classifier. All the features (except two – the predicate and position) require a full syntactic parse tree of the sentence for generating them. Three of the the features – the predicate lemma, voice and verb sub-categorization are shared by all the constituents in the the parse tree. Values of other features can differ across the constituents.

3.1.1 Classifier

For our experiments, we used TinySVM¹ along with YamCha² as the SVM training and test software (Kudo and Matsumoto, 2000, 2001). The SVM parameters such as the type of kernel and the values of various parameters was empirically determined using the development set. It was decided to use a polynomial kernel of degree 2; the cost per unit violation of the margin, $C=1$; and, tolerance of the termination criterion, $e=0.001$.

Support Vector Machines (SVMs) perform well on text classification tasks where data are represented in a high dimensional space using sparse feature vectors (Joachims, 1998; Lodhi et al., 2002). Inspired by the success of using SVMs for tagging syntactic chunks (Kudo and Matsumoto, 2000), we formulated the semantic role labeling problem as a multi-class classification problem using SVMs (Hacioglu et al., 2003; Pradhan et al., 2003b)

SVMs are inherently binary classifiers, but multi-class problems can be reduced to a number of binary-class problems using either the PAIRWISE approach or the ONE *vs* ALL (OVA) approach (Allwein et al., 2000). For a N class problem, in the PAIRWISE approach, a binary classifier is trained for each pair of the possible $\frac{N(N-1)}{2}$ class pairs. Whereas, in the OVA approach, N binary classifiers are trained to discriminate each class from a meta class created by combining the rest of the classes. Between these two approaches, there is a trade-off between the number of classifiers to be trained and the data used to train each classifier. While some experiments have been reported that the pairwise approach outperforms the OVA approach (Kressel, 1999), our initial experiments show better performance for OVA. Therefore we chose the OVA approach.

SVM outputs the distance of a feature vector from the maximum margin hyperplane. In order to facilitate probabilistic thresholding as well as generating an n -best

¹ <http://cl.aist-nara.ac.jp/~taku-ku/software/TinySVM/>

² <http://cl.aist-nara.ac.jp/~taku-ku/software/yamcha/>

hypotheses lattice, we convert the distances to probabilities by fitting a sigmoid to the scores as described in Platt (2000).

3.1.2 System Implementation

The system can be viewed as comprising two stages – the training stage and the testing stage. We will first discuss how the SVM is trained for this task. Since the training time taken by SVMs scales exponentially with the number of examples, and about 90% of the nodes in a syntactic tree have NULL argument labels, we found it efficient to divide the training process into two stages:

- (1) Filter out the nodes that have a very high probability of being NULL. A binary NULL *vs* NON-NULL classifier is trained on the entire dataset. A sigmoid function is fitted to the raw scores to convert the scores to probabilities as described by Platt (2000). All the training examples are run through this classifier and the respective scores for NULL and NON-NULL assignments are converted to probabilities using the sigmoid function. Nodes that are most likely NULL (probability > 0.90) are pruned from the training set. This reduces the number of NULL nodes by about 90% and the total number of nodes by about 80%. This is accompanied by a very negligible (about 1%) pruning of nodes that are NON-NULL.
- (2) The remaining training data are used to train OVA classifiers for all the classes along with a NULL class.

With this strategy only one classifier (NULL *vs* NON-NULL) has to be trained on all of the data. The remaining OVA classifiers are trained on the nodes passed by the filter (approximately 20% of the total), resulting in a considerable savings in training time.

In the testing stage, all the nodes are classified directly as NULL or one of the arguments using the classifier trained in step 2 above. We observe a slight decrease in recall if we filter the test examples using a NULL *vs* NON-NULL OVA classifier in a first

```

procedure SemanticParse(Sentence)
step Generate a full syntactic parse of the Sentence
step Identify all the verb predicates
for predicate  $\in$  Sentence do
    step Extract a set of features for each node in the tree relative
        to the predicate
    step Classify each feature vector using all OVA classifiers
    step Select the class of highest-scoring classifier
end for

```

Figure 3.1: The semantic role labeling algorithm

pass, as we do in the training process. This small performance gain is obtained at little or no cost of computation as SVMs are very fast in the testing phase. Pseudocode for the testing algorithm is shown in Figure 3.1.

3.1.3 Baseline System Performance

Table 3.1 shows the baseline performance numbers on all the three tasks mentioned earlier in Section 2.5.1 using the PropBank corpus with “hand-corrected” Penn Treebank parses³. The set features listed in Section 2.6 were used. These experiments are performed on the July 2002 release of PropBank. We followed the standard convention of using WSJ sections 02 to 21 for training, section 00 for development and section 23 for testing. We treat discontinuous and coreferential arguments in accordance to the CoNLL 2004 shared task on semantic role labeling. The first part of a discontinuous argument is labeled as it is, while the second part of the argument is labeled with a prefix “C-” appended to it. All coreferential arguments are labeled with a prefix “R-” appended to them. The training set comprises about 104,000 predicates instantiating about 250,000 arguments, and the test set comprises 5,400 predicates instantiating about 12,000 arguments.

For the argument identification and the combined identification and classification

³ Results on gold standard parses are reported because it is easier to compare with results from the literature; Results using actual errorful parser are reported in Table 6.7.

tasks, precision (P), recall (R) and the F_β ⁴ scores are reported, and for the argument classification task the classification accuracy (A) is reported.

	P	R	$F_{\beta=1}$	A
	(%)	(%)		(%)
Identification	91.2	89.6	90.4	
Classification	-	-	-	87.1
Identification + Classification	83.3	78.5	80.8	

Table 3.1: Baseline performance on all the three tasks using “gold-standard” parses

3.2 Improvements to ASSERT

3.2.1 Feature Engineering

Several new features were tested, two of which were obtained from the literature – named entities in constituents and head word part of speech. A χ^2 test of significance was used to determine whether the difference in number of responses over all the confusion categories (correct, wrong, false positive and false negative) are statistically significant at $p = 0.05$. All the statistically significant improvements are marked with an *.

3.2.1.1 Named Entities in Constituents

Surdeanu et al. (2003) reported a performance improvement on classifying the semantic role of the constituent by using the presence of a named entity in the constituent. It is expected that some of these entities such as location and time are particularly important for the adjunctive arguments ARGM-LOC and ARGM-TMP. Entity tags should also help in cases where the head words are not common, or for a closed set of locative or temporal cues, like “in *Mexico*”, or “in *2003*”, etc. A similar use of named entities was evaluated in our system. We tagged 7 named entities (PERSON, ORGANIZATION, LOCATION,

⁴ $F_\beta = \frac{(\beta^2+1) \times P \times R}{\beta^2 \times P + R}$

PERCENT, MONEY, TIME, DATE) in the corpus using *IdentiFinder* (Bikel et al., 1999) and added them 7 binary features. Each of these features is true if the respective named entity is contained in the constituent.

3.2.1.2 Head Word Part of Speech

Surdeanu et al. (2003) showed that adding part of speech of the head word of a constituent as a feature in the task of argument identification gave a significant performance boost to their decision tree based system.

Two other ways of generalizing the head word were tried: i) adding the head word cluster as a feature, and ii) replacing the head word with a named entity if it belonged to any of the seven named entities mentioned earlier.

3.2.1.3 Verb Clustering

The predicate is one of the most salient features in predicting the argument class. Since our training data is relatively limited, any real world test set will contain predicates that have not been seen in training. In these cases, we can benefit from some information about the predicate by creating clusters or classes and using them as features.

The distance function used for clustering is based on the intuition that verbs with similar semantics will tend to have similar direct objects. For example, verbs such as “eat”, “devour”, “savor”, will tend to all occur with direct objects describing food. The clustering algorithm uses a database of verb-direct-object relations extracted by Lin (1998a). The verbs were clustered into 64 classes using the probabilistic co-occurrence model of Hofmann and Puzicha (1998). We then use the verb class of the current predicate as a feature.

3.2.1.4 Generalizing the Path Feature

As will be seen in Section 3.2.7.2, for the argument identification task, path is one of the most salient features. However, it is also the most data sparse feature. To overcome this problem, the path was generalized in three different ways:

- (1) Compressing sequences of identical labels into one following the intuition that successive embedding of the of the same phrase in the tree might not add additional information.
- (2) Removing the direction in the path, thus making insignificant the point at which it changes direction in the tree, and
- (3) Using only that part of the path from the constituent to the lowest common ancestor of the predicate and the constituent – “Partial Path”. For example, the partial path for the path illustrated in Figure 2.11 is NP↑S.

3.2.1.5 Oracle Verb Sense Information

The arguments that a predicate can take depend on the sense of the predicate. Each predicate tagged in the PropBank corpus is assigned a separate set of arguments depending on the sense in which it is used. Table 3.2 illustrates the argument sets for the word. Depending on the sense of the predicate *talk*, either ARG1 or ARG2 can identify the *hearer*. Absence of this information can be potentially confusing to the learning mechanism.

We added the oracle sense information extracted from PropBank, to our features by treating each sense of a predicate as a distinct predicate.

3.2.1.6 Noun Head of Prepositional Phrases

Many adjunctive arguments, such as temporals and locatives, occur as prepositional phrases in a sentence, and it is often the case that the head words of those phrases,

Talk	sense 1: speak		sense 2: persuade/dissuade	
Tag	Description	Tag	Description	
ARG0	Talker	ARG0	Talker	
ARG1	Subject	ARG1	Talked to	
ARG2	Hearer	ARG2	Secondary action	

Table 3.2: Argument labels associated with the two senses of predicate *talk* in PropBank corpus.

which are always prepositions, are not very discriminative, eg., “in the city”, and “in a few minutes” both share the same head word “in” and neither contain a named entity, but the former is ARG_M-LOC, whereas the latter is ARG_M-TMP. Therefore we replaced the head word of a prepositional phrase with that of the first noun phrase inside the prepositional phrase. The preposition information was retained by appending it to the phrase type, eg., in Figure 3.2 the head word of the phrase “for about 20 minutes” was originally the preposition “for”. After the transformation, the head word was changed to “minu

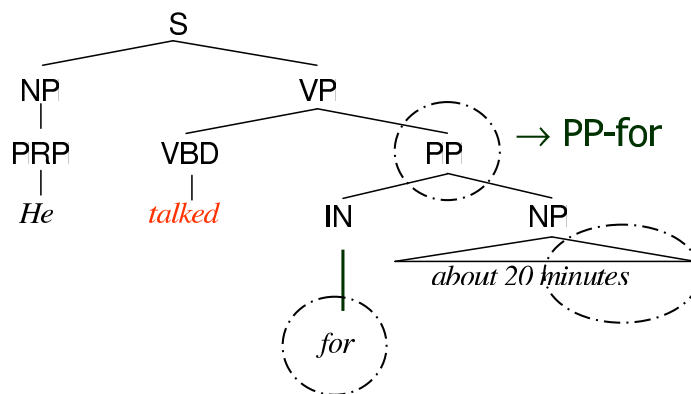


Figure 3.2: Noun head of PP.

3.2.1.7 First and Last Word/POS in Constituent

Some arguments tend to contain discriminative first and last words so we tried using them along with their part of speech as four new features.

3.2.1.8 Ordinal Constituent position

In order to avoid false positives where constituents far away from the predicate are spuriously identified as arguments, we added this feature which is a concatenation of the constituent type and its ordinal position from the predicate. Figure 3.3 shows the different ordinal positions for phrases in a sample tree.

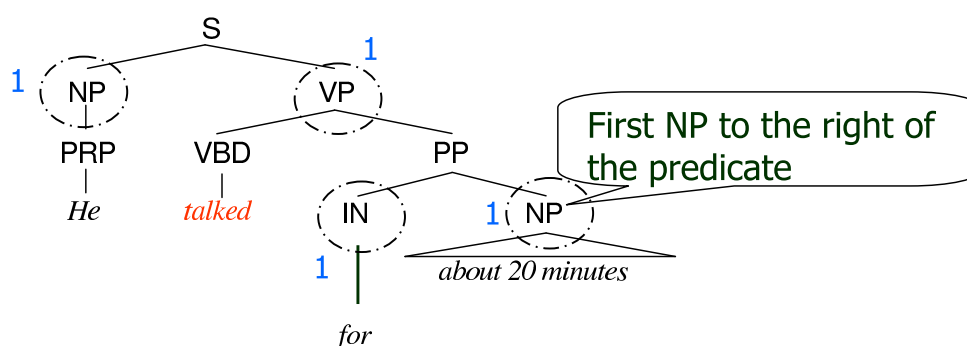


Figure 3.3: Ordinal constituent position.

3.2.1.9 Constituent Tree Distance

This is a finer way of specifying the already present position feature. Figure 3.4 shows a sample tree with the tree distances for each constituent.

3.2.1.10 Constituent Relative Features

These are nine features representing the constituent type, head word and head word part of speech of the parent, and left and right siblings of the constituent in focus,

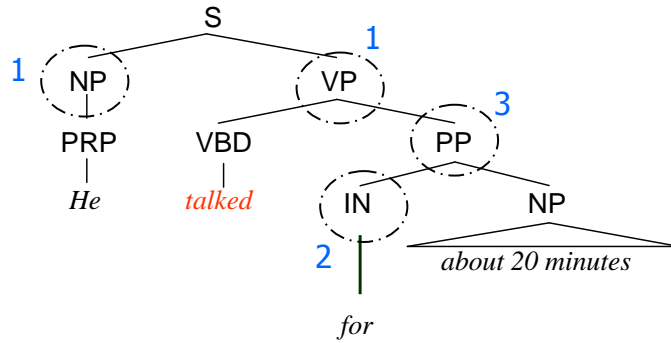


Figure 3.4: Tree distance.

as shown in Figure 3.5. These were added on the intuition that encoding the tree context this way might add robustness and improve generalization.

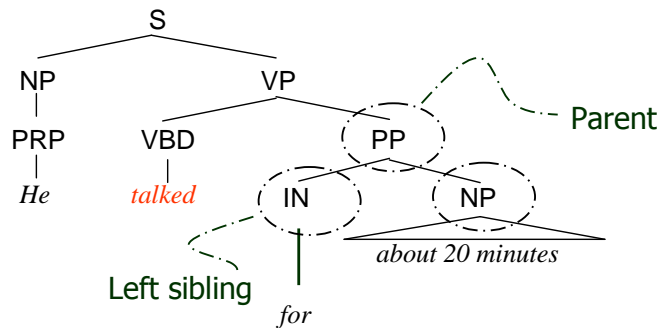


Figure 3.5: Sibling features.

3.2.1.11 Temporal Cue Words

There are several temporal cue words that are not captured by the named entity tagger and were added as binary features indicating their presence. The BOW toolkit was used to identify words and bigrams that had highest average mutual information with the ARGM-TMP argument class. The complete list of words is given in Appendix A.

3.2.1.12 Dynamic Class Context

In the task of argument classification, these are dynamic features that represent the hypotheses of at most previous two non-null nodes belonging to the same tree as the node being classified.

3.2.1.13 CCG Parse Features

While the Path feature has been identified to be very important for the argument identification task, it is one of the most sparse features and may be difficult to train or generalize (Pradhan et al., 2004; Xue and Palmer, 2004). A dependency grammar should generate shorter paths from the predicate to dependent words in the sentence, and could be a more robust complement to the phrase structure grammar paths extracted from the PSG parse tree. Gildea and Hockenmaier (2003) report that using features extracted from a Combinatory Categorical Grammar (CCG) representation improves semantic labeling performance on core arguments (ARG0-5). We evaluated features from a CCG parser – StatCCG, combined with our baseline feature set. Figure 3.6 shows the CCG parse of the sentence “London denied plans on Monday.” We used three features that were introduced by Gildea and Hockenmaier (2003):

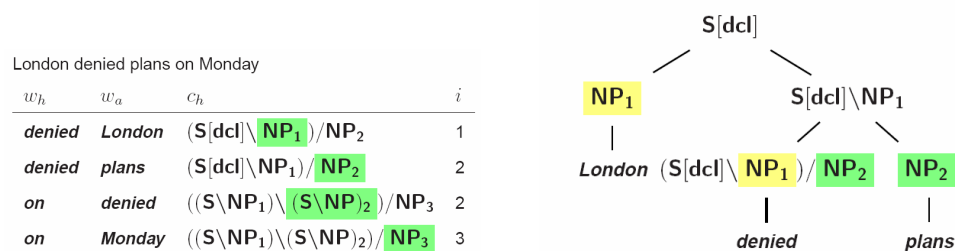


Figure 3.6: CCG parse.

Phrase type – This is the category of the maximal projection between the two words – the predicate and the dependent word.

Categorial Path – This is a feature formed by concatenating the following three values: i) category to which the dependent word belongs, ii) the direction of dependence and iii) the slot in the category filled by the dependent word. For example, for the tree in Figure 3.6 the path between *denied* and *plans* would be (S[dc] \ NP) / NP.2. ←

Tree Path – This is the categorial analogue of the path feature in the Charniak parse based system, which traces the path from the dependent word to the predicate through the binary CCG tree.

3.2.1.14 Path Generalizations

Clause-based path variations – Position of the clause node (S, SBAR) seems to be an important feature in argument identification (Hacioglu et al., 2004) Therefore we experimented with four clause-based path feature variations.

- Replacing all the nodes in a path other than clause nodes with an “*”. For example, the path NP↑S↑VP↑SBAR↑NP↑VP↓VBD becomes
NP↑S↑*S↑*↑*↓VBD
- Retaining only the clause nodes in the path, which for the above example would produce NP↑S↑S↓VBD,
- Adding a binary feature that indicates whether the constituent is in the same clause as the predicate,
- Collapsing the nodes between S nodes which gives NP↑S↑NP↑VP↓VBD.

Path n-grams – This feature decomposes a path into a series of trigrams. For example, the path NP↑S↑VP↑SBAR↑NP↑VP↓VBD becomes: NP↑S↑VP, S↑VP↑SBAR, VP↑SBAR↑NP, SBAR↑NP↑VP, etc. Shorter paths were padded with nulls.

Single character phrase tags – Each phrase category is clustered to a category defined by the first character of the phrase label.

3.2.1.15 Predicate Context

We added the predicate context to capture predicate sense variations. Two words before and two words after were added as features. The POS of the words were also added as features.

3.2.1.16 Punctuation

For some adjunctive arguments, punctuation plays an important role so we added punctuation immediately before and after the constituent as new features.

3.2.1.17 Feature Context

Features of constituents that are parent or siblings of the constituent being classified were found useful. In the current machine learning technique, we classify each of the constituents independently, however, in reality, there is a complex interaction between the types and number of arguments that a constituent can assume, given classifications of other nodes. As we will look at later, we perform a post-processing step using the argument sequence information, but that does not cover all possible constraints. One way of trying to capture those best in the current architecture would be to take into consideration the feature vector compositions of all the NON-NULL constituents for the sentence. This is exactly what this feature does. It uses all the other feature vector values of the constituents that have been found to be likely NON-NULL, as an added context.

3.2.2 Feature Selection and Calibration

In the baseline system, we used the same set of features for all the n binary ONE *vs* ALL classifiers. While performing some ablation studies we found that adding the named entity features to NULL *vs* NON-NULL classifier had a detrimental effect on the performance on the task of argument identification, however, the same feature set showed significant improvement to the task of argument classification. Therefore, we thought that optimizing subsets of features for each argument class might improve performance. To achieve this, we performed a simple feature selection procedure. For each classifier, we started with the set of features introduced by Gildea and Jurafsky (2002). We pruned this set by training classifiers after leaving out one feature at a time and checking its performance on a development set. We used the χ^2 test of significance while making pruning decisions. Following that, we added each of the other features one at a time to the pruned baseline set of features and selected ones that showed significantly improved performance. Since the feature selection experiments were computationally intensive, we performed them using 10k randomly selected training examples.

SVMs output distances not probabilities. These distances may not be comparable across classifiers, especially if different features are used to train each binary classifier. In the baseline system, we used the algorithm described by Platt (2000) to convert the SVM scores into probabilities by fitting to a sigmoid. When all classifiers used the same set of features, fitting all scores to a single sigmoid was found to give the best performance. Since different feature sets are now used by the classifiers, we trained a separate sigmoid for each classifier.

Foster and Stine (Foster and Stine, 2004) show that the pool-adjacent-violators (PAV) algorithm (Barlow et al., 1972) provides a better method for converting raw classifier scores to probabilities when Platt’s algorithm fails. The probabilities resulting from either conversions may not be properly calibrated. So, we binned the probabilities

	Raw Scores	Probabilities	
		After lattice-rescoring	
		Uncalibrated	Calibrated
	(%)	(%)	(%)
Same Feat. same sigmoid	74.7	74.7	75.4
Selected Feat. diff. sigmoids	75.4	75.1	76.2

Table 3.3: Performance improvement on selecting features per argument and calibrating the probabilities on 10k training data.

and trained a warping function to calibrate them. For each argument classifier, we used both the methods for converting raw SVM scores into probabilities and calibrated them using a development set. Then, we visually inspected the calibrated plots for each classifier and chose the method that showed better calibration as the calibration procedure for that classifier. Plots of the predicted probabilities versus true probabilities for the ARGUMENT vs ALL classifier, before and after calibration are shown in Figure 3.7. Ideally the predicted probability should be equal to the true probability, and so all the points on the graph should fall on a 45 degree line passing through the origin. From the graph before calibration, it can be seen that the system was underestimating the probability, and after calibration part of the region from 0-70% was getting slightly overestimated, where as the one from 70-100% range was getting calibrated quite well, giving an overall performance improvement. It might be the case that the fact that some part of the probability range was getting overestimated was not as important towards the joint classification accuracy. The performance improvement over a classifier that is trained using all the features for all the classes is shown in Table 3.3.

3.2.3 Disallowing Overlaps

The system as described above might label two constituents even if they overlap in words. Since we are dealing with parse tree, this can only occur as a subsumption as in the following example:

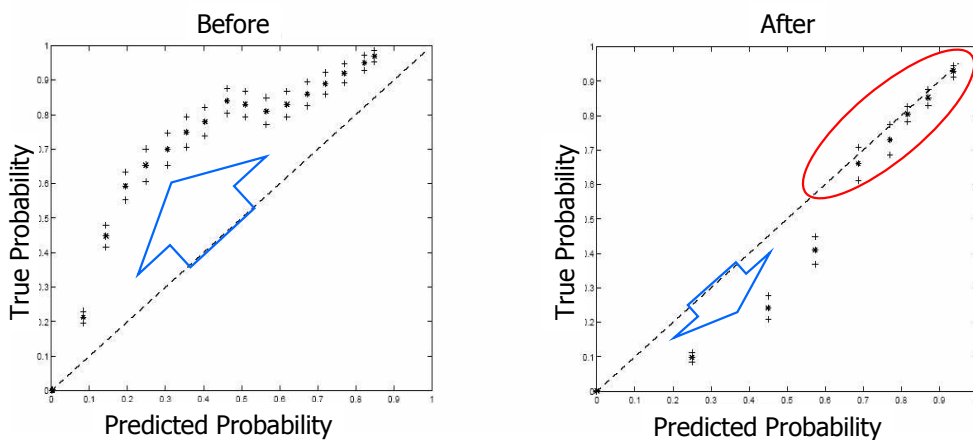


Figure 3.7: Plots showing true probabilities versus predicted probabilities before and after calibration on the test set for ARGM-TMP

But [*ARG0* nobody] [*predicate* knows] [*ARG1* at what level [*ARG1* the futures and stocks will open today]]

This is a problem, since overlapping arguments were not allowed in PropBank. The system chooses among overlapping constituents by retaining the one for which the SVM has the highest confidence, and labeling the others NULL. Since we are dealing with strictly hierarchical trees, nodes overlapping in words always have an ancestor-descendant relationship, and therefore the overlaps are restricted to subsumptions only. The probabilities obtained by applying the sigmoid function to the raw SVM scores are used as the measure of confidence. Table 3.4 shows the performance improvement on the task of identifying and labeling semantic arguments using Treebank parses. In this system, the overlap-removal decisions are taken independently of each other.

	P	R	$F_{\beta=1}$
	(%)	(%)	
Baseline	83.3	78.5	80.8
No Overlaps	85.4	78.1	*81.6

Table 3.4: Improvements on the task of argument identification and classification after disallowing overlapping constituents.

3.2.4 Argument Sequence Information

In order to improve the performance of their statistical argument tagger, Gildea and Jurafsky used the fact that a predicate is likely to instantiate a certain set of argument types. We use a similar strategy, with some additional constraints: i) argument ordering information is retained, and ii) the predicate is considered as an argument and is part of the sequence. We achieve this by training a trigram language model on the argument sequences, so unlike Gildea and Jurafsky, we can also estimate the probability of argument sets not seen in the training data. We first convert the raw SVM scores to probabilities as described earlier. Then, for each sentence being parsed, we generate an argument lattice using the n -best hypotheses for each node in the syntax tree. We then perform a Viterbi search through the lattice using the probabilities assigned by the sigmoid as the observation probabilities, along with the language model probabilities, to find the maximum likelihood path through the lattice, such that each node is either assigned a value belonging to the PROPBANK ARGUMENTS, or NULL.

CORE ARGS/ HAND-CORRECTED PARSES	P	R	F_1
Baseline w/o overlaps	90.5	87.4	88.9
Common predicate	91.2	86.9	89.0
Specific predicate lemma	91.7	87.2	*89.4

Table 3.5: Improvements on the task of argument identification and classification using Treebank parses, after performing a search through the argument lattice.

The search is constrained in such a way that no two NON-NULL nodes overlap

with each other. To simplify the search, we allowed only NULL assignments to nodes having a NULL likelihood above a threshold. While training the language model, we can either use the actual predicate to estimate the transition probabilities in and out of the predicate, or we can perform a joint estimation over all the predicates. We implemented both cases considering two best hypotheses, which always includes a NULL (we add NULL to the list if it is not among the top two). On performing the search, we found that the overall performance improvement was not much different than that obtained by resolving overlaps as mentioned earlier. However, we found that there was an improvement in the CORE-ARGUMENTS accuracy on the combined task of identifying and assigning semantic arguments, given Treebank parses, whereas the accuracy of the ADJUNCTIVE ARGUMENTS slightly deteriorated. This seems to be logical considering the fact that the ADJUNCTIVE ARGUMENTS have looser constraints on their ordering and even the quantity. We therefore decided to use this strategy only for the CORE ARGUMENTS. Although, there was an increase in F_1 score when the language model probabilities were jointly estimated over all the predicates, this improvement is not statistically significant. However, estimating the same using specific predicate lemmas, showed a significant improvement in accuracy. The performance improvement is shown in Table 3.5.

3.2.5 Alternative Pruning Strategies

SVMs are costly in terms of the time they take to train. We tried to find out some filtering strategy that could be employed, and that reduced the training time significantly without having a significant impact on the classification accuracy. We call the strategy that uses all the data for training as the “One-pass no-prune” strategy. The three strategies in order of decreasing training time are:

- **One-pass no-prune** strategy that uses all the data to train ONE *vs* ALL classifiers
- one for each argument including the NULL argument. This has considerably

higher training time as compared to the other two.

Two-pass soft-prune strategy (baseline), uses a NULL *vs* NON-NULL classifier trained on the entire data, filters out nodes with high confidence (probability > 0.9) of being NULL in a first pass and then trains ONE *vs* ALL classifiers on the remaining data *including* the NULL class.

Two-pass hard-prune strategy, which uses a NULL *vs* NON-NULL classifier trained on the entire data in a first pass. All nodes labeled NULL are filtered. And then, ONE *vs* ALL classifiers are trained on the data containing *only* NON-NULL examples. There is no NULL *vs* ALL classifier in the second stage.

Table 3.6 shows performance on the task of identifying and labeling PropBank arguments. There is no statistically significant difference between the *two-pass soft-prune* strategy, and the *one-pass no-prune* strategy. However, both are better than the *two-pass hard-prune* strategy. Our initial choice of training strategy was dictated by the following efficiency considerations: i) SVM training is a convex optimization problem that scales exponentially with the size of the training set, ii) On an average about 90% of the nodes in a tree are NULL arguments, and iii) Only one classifier has to optimized on the entire data. The *two-pass soft-prune* strategy was continued as before.

	No Overlaps		
	P (%)	R (%)	$F_{\beta=1}$
Two-pass hard-prune	84	80	*81.9
Two-pass soft-prune	86	81	83.4
One-pass no-prune	87	80	83.3

Table 3.6: Comparing pruning strategies

3.2.6 Best System Performance

It was found that a subset of the features helped argument identification and another subset helped argument classification. We will look at the individual feature contribution in Section 3.2.7.1. The best system is trained by first filtering the most likely nulls using the best NULL *vs* NON-NULL classifier trained using all the features whose argument identification F_1 score is marked in bold in Table 3.9, and then training a ONE *vs* ALL classifier using the data remaining after performing the filtering and using the features that contribute positively to the classification task – ones whose accuracies are marked in bold in Table 3.9. Table 3.7 shows the performance of this system.

CLASSES	TASK	HAND-CORRECTED PARSES			
		P	R	F_1	A
ALL	Id.	95.2	92.5	93.8	
ARGS	Classification	-	-	-	91.0
	Id. + Classification	88.9	84.6	86.7	
CORE	Id.	96.2	93.0	94.6	
ARGS	Classification	-	-	-	93.9
	Id. + Classification	90.5	87.4	88.9	

Table 3.7: Best system performance on all three tasks using Treebank parses.

Owing to the Feb 2004 release of much more and completely adjudicated PropBank data, we have a chance to report our performance numbers on this data set. Table 3.19 shows the same information as in previous Tables 3.7, but generated using the new data.

ALL ARGS	TASK	P	R	F_1	A
HAND	Id.	96.2	95.8	96.0	
	Classification	-	-	-	93.0
	Id. + Classification	89.9	89.0	89.4	

Table 3.8: Best system performance on all three tasks on the latest PropBank data, using Treebank parses.

3.2.7 System Analysis

3.2.7.1 Feature Performance

Table 3.9 shows the effect each feature has on the argument classification and argument identification tasks, when added individually to the baseline. Addition of named entities to the NULL *vs* NON-NULL classifier degraded its performance. We attribute this to a combination of two things: i) there are a significant number of constituents that contain named entities, but are not arguments of a predicate (the parent of an argument node will also contain the same named entity) therefore this provides a noisy feature for NULL *vs* NON-NULL classification. and ii) SVMs don't seem to handle irrelevant features very well (Weston et al., 2001). We then tried using this as a feature solely in the task of classifying constituents known to represent arguments, using features extracted from Treebank parses. On this task, overall classification accuracy increased from 87.9% to 88.1%. As expected, the most significant improvement was for adjunct arguments like temporals (ARGM-TMP) and locatives (ARGM-LOC) as shown in Table 3.10. Since the number of locative and temporal arguments in the test set is quite low (< 10%) as compared to the core arguments, the increase in performance on these does not boost the overall performance significantly.

Adding head word POS as a feature significantly improves both the argument classification and the argument identification tasks. We tried two other ways of generalizing the head word: i) adding the head word cluster as a feature, and ii) replacing the head word with a named entity if it belonged to any of the seven named entities mentioned earlier. Neither method showed significant improvement. Table 3.9 also shows the contribution of replacing the head word and the head word POS separately in the feature where the head of a prepositional phrase is replaced by the head word of the noun phrase inside it. A combination of relative features seem to have a significant improvement on either or both the classification and identification tasks, and so do the

first and last words in the constituent. None of the path generalizations except the features significantly affected NULL *vs* NON-NULL classification. The “Partial-Path” feature improved the argument classification performance for “gold-standard” parses from 87.9% to 88.3%.

3.2.7.2 Feature Saliency

In analyzing the performance of the system, it is useful to estimate the relative contribution of the various feature sets used. Table 3.11 shows the argument classification accuracies for combinations of features on the training and test set for all PropBank arguments, using Treebank parses.

In the upper part of Table 3.11 we see the degradation in performance by leaving out one feature at a time. The features are arranged in the order of increasing saliency. Removing all head word related information has the most detrimental effect on the performance. The lower part of the table shows the performance of some feature combinations by themselves.

Table 3.12 shows the feature saliency on the task of argument identification. As opposed to the argument classification task, where removing the path has the least effect on performance, on the task of argument identification, removing the path causes the convergence in SVM training to be very slow and has the most detrimental effect on the performance.

FEATURES	ARGUMENT		ARGUMENT		
	CLASSIFICATION		IDENTIFICATION		
	A	P	R	F ₁	
Baseline	87.9	93.7	88.9	91.3	
+ Named entities	88.1	93.3	88.9	91.0	
+ Head POS	* 88.6	94.4	90.1	* 92.2	
+ Verb cluster	88.1	94.1	89.0	91.5	
+ Partial path	88.2	93.3	88.9	91.1	
+ Verb sense	88.1	93.7	89.5	91.5	
+ Noun head PP (only POS)	* 88.6	94.4	90.0	* 92.2	
+ Noun head PP (only head)	* 89.8	94.0	89.4	91.7	
+ Noun head PP (both)	* 89.9	94.7	90.5	* 92.6	
+ First word in constituent	* 89.0	94.4	91.1	* 92.7	
+ Last word in constituent	* 89.4	93.8	89.4	91.6	
+ First POS in constituent	88.4	94.4	90.6	* 92.5	
+ Last POS in constituent	88.3	93.6	89.1	91.3	
+ Ordinal const. pos. concat.	87.7	93.7	89.2	91.4	
+ Const. tree distance	88.0	93.7	89.5	91.5	
+ Parent constituent	87.9	94.2	90.2	* 92.2	
+ Parent head	85.8	94.2	90.5	* 92.3	
+ Parent head POS	* 88.5	94.3	90.3	* 92.3	
+ Right sibling constituent	87.9	94.0	89.9	91.9	
+ Right sibling head	87.9	94.4	89.9	* 92.1	
+ Right sibling head POS	88.1	94.1	89.9	92.0	
+ Left sibling constituent	* 88.6	93.6	89.6	91.6	
+ Left sibling head	86.9	93.9	86.1	89.9	
+ Left sibling head POS	* 88.8	93.5	89.3	91.4	
+ Temporal cue words	* 88.6	-	-	-	
+ Dynamic class context	88.4	-	-	-	

Table 3.9: Effect of each feature on the argument classification task and argument identification task, when added to the baseline system.

Table 3.10: Improvement in classification accuracies after adding named entity information.

	ARGM-LOC			ARGM-TMP		
	P	R	F ₁	P	R	F ₁
Baseline	61.8	56.4	59.0	76.4	81.4	78.8
With Named Entities	70.7	67.0	*68.8	81.1	83.7	*82.4

FEATURES	ACCURACY
<i>All</i>	91.0
<i>All except Path</i>	90.8
<i>All except Phrase Type</i>	90.8
<i>All except HW and HW-POS</i>	90.7
<i>All except All Phrases</i>	*83.6
<i>All except Predicate</i>	*82.4
<i>All except HW and FW and LW info.</i>	*75.1
<i>Only Path and Predicate</i>	74.4
<i>Only Path and Phrase Type</i>	47.2
<i>Only Head Word</i>	37.7
<i>Only Path</i>	28.0

Table 3.11: Performance of various feature combinations on the task of argument classification.

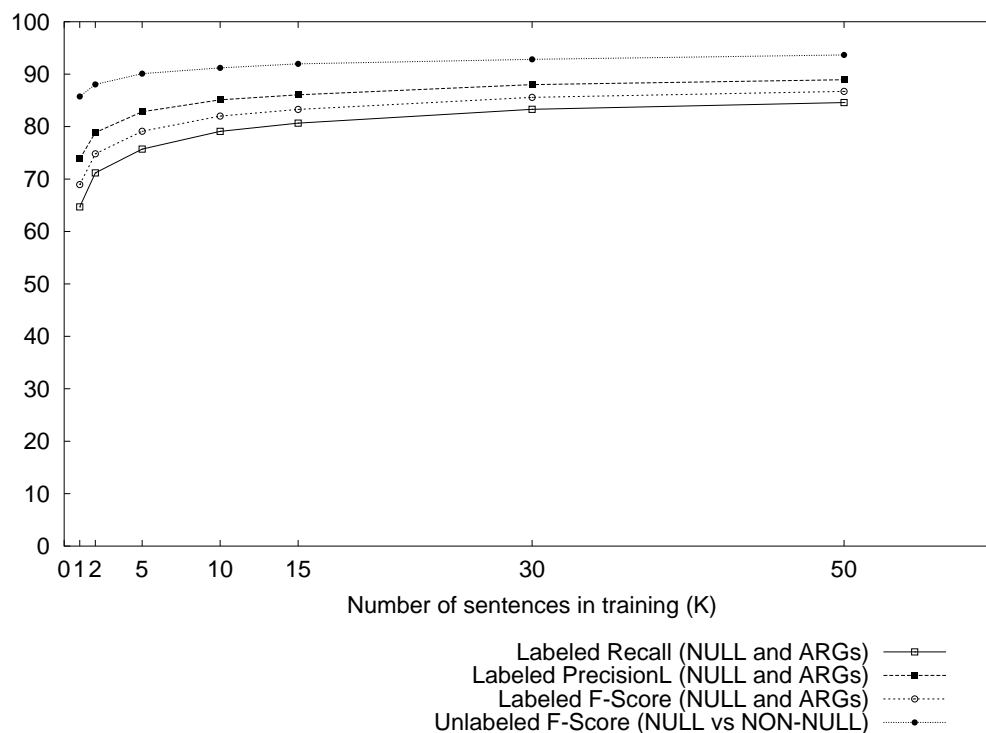


Figure 3.8: Learning curve for the task of identifying and classifying arguments using Treebank parses.

FEATURES	P	R	F ₁
<i>All</i>	95.2	92.5	93.8
<i>All except HW</i>	95.1	92.3	93.7
<i>All except Predicate</i>	94.5	91.9	93.2
<i>All except HW and FW and LW info.</i>	91.8	88.5	*90.1
<i>All except Path and Partial Path</i>	88.4	88.9	*88.6
<i>Only Path and HW</i>	88.5	84.3	86.3
<i>Only Path and Predicate</i>	89.3	81.2	85.1

Table 3.12: Performance of various feature combinations on the task of argument identification

3.2.7.3 Size of Training Data

One important concern in any supervised learning method is the amount of training examples required for the near optimum performance of a classifier. To check the behavior of this learning problem, we trained the classifiers on varying amounts of training data. The resulting plots are shown in Figure 3.8. We approximate the curves by plotting the accuracies at seven data sizes. The first curve from the top indicates the change in F₁ score on the task of argument identification alone. The third curve indicates the F₁ score on the combined task of argument identification and classification. It can be seen that after about 10k examples, the performance starts asymptotic, which indicates that simply tagging more data might not be a good strategy. What needs to be done is to try and tag appropriate new data. Also, the fact that the first and third curves – first being the F-score on the task of argument identification and the third being the F-score on the combined task of identification and classification – run almost parallel to each other tells us that there is a constant loss due to classification errors throughout the data range. One way to bridge this gap could be to identify better features. In order to get a good approximation across all predicates, the training data was accumulated by selecting examples at random. We also plotted the precision and recall values for the combined task of identification and classification.

3.2.7.4 Performance Tuning

The expectations of a semantic role labeler by a real-world application would vary considerably. Some applications might favor precision over recall, and others vice-versa. Since we have a mechanism of assigning confidence to the hypothesis generated by the labeler, we can tune its performance accordingly. Table 3.13 shows the achievable increase in precision with some drop in recall for the task of identifying and classifying semantic arguments using the Treebank parses. The arguments that get assigned probability below the confidence threshold are assigned a null label.

CONFIDENCE THRESHOLD	P	R
0.10	88.9	84.6
0.25	89.0	84.6
0.50	91.7	81.4
0.75	96.7	65.2
0.90	98.7	32.3
0.95	98.5	12.3

Table 3.13: Precision/Recall table for the combined task of argument identification and classification using Treebank parses.

3.2.8 Comparing Performance with Other Systems

Our system is compared against 4 other semantic role labelers in the literature. In comparing systems, results are reported for all the three types of tasks mentioned earlier.

3.2.8.1 Comparing Classifiers

Since two systems, in addition to ours, report results using the same set of features on the same data, the influence of the classifiers can be directly assessed. G&P system estimates the posterior probabilities using several different feature sets and interpolates the estimates, while Surdeanu et al. (2003) use a decision tree classifier. Table 3.14

Classifier	Accuracy (%)
SVM	88
Decision Tree (Surdeanu et al., 2003)	79
Gildea and Palmer (2002)	77

Table 3.14: Argument classification using same features but different classifiers.

shows a comparison between the three systems for the task of argument classification. Keeping the same features and changing only the classifier produces a 9% absolute increase in performance on the same test set.

3.2.8.2 Argument Identification (NULL *vs* NON-NULL)

Table 3.15 compares the results of the task of identifying the parse constituents that represent semantic arguments.

Classes	System	Treebank Parses		
		P	R	F ₁
ARG0-5	ASSERT	95	91	93
+	Surdeanu System II	-	-	89
ARGMS	Surdeanu System I	85	84	85

Table 3.15: Argument identification

3.2.8.3 Argument Classification

Table 3.16 compares the argument classification accuracies of various systems, and at various levels of classification granularity, and parse accuracy. It can be seen that ASSERT performs significantly better than all the other systems on all PropBank arguments.

Classes	System	Treebank parses
		Accuracy
ARG0-5	ASSERT	90
+	G&P	77
ARGMS	Surdeanu System II	84
	Surdeanu System I	79
CORE ARGS	ASSERT	93.9
(ARG0-5)	C&R System II	93.5
	C&R System I	92.4

Table 3.16: Argument classification

3.2.8.4 Argument Identification and Classification

Table 3.17 shows the results for the task where the system first identifies candidate argument boundaries and then labels them with the most likely argument. This is the hardest of the three tasks outlined earlier. ASSERT does a very good job of generalizing in both stages of processing.

Classes	System	Treebank parses		
		P	R	F ₁
ARG0-5	ASSERT	86	83	85
+	G&H System I	76	68	72
ARGMS	G&P	71	64	67
ARG0-5	ASSERT	89	86	87
	G&H System I	82	79	80
	C&R System II	-	-	-

Table 3.17: Argument Identification and classification

3.2.9 Using Automatically Generated Parses

Thus far, all reported results were using Treebank parses. In real-world applications, ASSERT will have to extract features from an automatically generated parse. To evaluate this scenario, the Charniak parser (Charniak, 2000) was used to generate parses for PropBank training and test data. The predicates were lemmatized automat-

ically using the XTAG morphology database⁵ (Daniel et al., 1992). Table 6.7 shows the performance degradation when errorfull automatically generated parses are used.

Classes	Task	Automatic parses			
		P	R	$F_{\beta=1}$	A
		(%)	(%)		(%)
ALL	Id.	89.6	80.1	84.6	
ARGS	Classification	-	-	-	89.6
	Id. + Classification	81.6	73.3	76.9	
CORE	Id.	90.7	82.9	86.6	
ARGS	Classification	-	-	-	90.8
	Id. + Classification	84.5	77.2	80.7	

Table 3.18: Performance degradation when using automatic parses instead of Treebank ones.

Table 3.19 shows the performance of the system after adding the CCG features, additional features extracted from the Charniak parse tree, and performing feature selection and calibration. Numbers in parentheses are the corresponding baseline performances.

3.2.10 Comparing Performance with Other Systems

3.2.10.1 Argument Identification

Table 3.20 shows comparative performance on the task of argument identification.

As expected, the performance degrades considerably when features are extracted from an automatic parse as opposed to a Treebank parse. This indicates that the syntactic parser performance directly influences the argument boundary identification performance. This could be attributed to the fact that the two features, viz., Path and Head Word that have been seen to be good discriminators of the semantically salient nodes in the syntax tree, are derived from the syntax tree.

⁵ <ftp://ftp.cis.upenn.edu/pub/xtag/morph-1.5/morph-1.5.tar.gz>

TASK	P	R	F ₁	A
	(%)	(%)		(%)
Id.	86.9 (86.8)	84.2 (80.0)	85.5 (83.3)	
Class.	-	-	-	92.0 (90.1)
Id. + Class.	82.1 (80.9)	77.9 (76.8)	79.9 (78.8)	

Table 3.19: Best system performance on all tasks using automatically generated syntactic parses.

3.2.10.2 Argument Classification

Table 3.21 shows comparative performance on the task of argument classification.

3.2.10.3 Argument Identification and Classification

Table 3.22 shows the comparative performance on the combined task of argument identification and classification.

Classes	System	Automatic		
		P	R	F ₁
ARG0-5	ASSERT	90	80	85
+	Surdeanu System II	-	-	-
ARGMS	Surdeanu System I	-	-	-

Table 3.20: Argument identification

Classes	System	Automatic
		Accuracy
ARG0-5	ASSERT	90
+	G&P	74
ARGMS	Surdeanu System II	-
	Surdeanu System I	-
CORE ARGS	ASSERT	91
(Arg0-5)	C&R System II	-
	C&R System I	-

Table 3.21: Argument classification

Classes	System	Automatic		
		P	R	F ₁
ARG0-5	ASSERT	82	73	77
+	G&H System I	71	63	67
ARGMS	G&P	58	50	54
ARG0-5	ASSERT	84	77	81
	G&H System I	76	73	75
	C&R System II	65	75	70

Table 3.22: Argument Identification and classification

3.3 Labeling Text from a Different Corpus

Thus far, all experiments our unseen test data was the section 23 of WSJ. In other words, it represented the same text source as the training data. In order to see how well the features generalize to texts drawn from a different text source, the classifier trained on PropBank training data was used to test data drawn from two different corpora – The AQUAINT corpus, and the BROWN corpus.

3.3.1 AQUAINT Test Set

The AQUAINT corpus (LDC, 2002) is a collection of text from the New York Times Inc., Associated Press Inc., and Xinhua News Service (whereas, PropBank represents Wall Street Journal data). About 400 sentences were annotated from the AQUAINT corpus with PropBank arguments. The results are shown in Table 3.23.

Task		P (%)	R (%)	$F_{\beta=1}$	A
ALL	Id.	74.4	71.1	72.7	
ARGS	Classification	-	-	-	82.9
	Id. + Classification	65.4	60.4	62.8	
CORE	Id.	87.0	72.9	79.3	
ARGS	Classification	-	-	-	83.9
	Id. + Classification	78.6	65.9	71.7	

Table 3.23: Performance on the AQUAINT test set.

There is a significant drop in the precision and recall numbers for the AQUAINT test set (compared to the precision and recall numbers for the PropBank test set which were 82% and 78% respectively). One possible reason for the drop in performance is relative coverage of the features on the two test sets. The head word, path and predicate features all have a large number of possible values and could contribute to lower coverage when moving from one domain to another. Also, being more specific they might not transfer well across domains.

Features	Arguments (%)	non-Arguments (%)
<i>Predicate, Path</i>	87.60	2.91
<i>Predicate, Head Word</i>	48.90	26.55
<i>Cluster, Path</i>	96.31	4.99
<i>Cluster, Head Word</i>	83.85	60.14
<i>Path</i>	99.13	15.15
<i>Head Word</i>	93.02	90.59

Table 3.24: Feature Coverage on PropBank test set using semantic role labeler trained on PropBank training set.

Features	Arguments (%)	non-Arguments (%)
<i>Predicate, Path</i>	62.11	4.66
<i>Predicate, Head Word</i>	30.26	17.41
<i>Cluster, Path</i>	87.19	10.68
<i>Cluster, Head Word</i>	65.82	45.43
<i>Path</i>	96.50	29.26
<i>Head Word</i>	84.65	83.54

Table 3.25: Coverage of features on AQUAINT test set using semantic role labeler trained on PropBank training set.

Table 3.24 shows the coverage for features on the PropBank test set. The tables show feature coverage for constituents that were Arguments and constituents that were NULL. About 99% of the predicates in the AQUAINT test set were seen in the PropBank training set. Table 3.25 shows coverage for the same features on the AQUAINT test set.

Chapter 4

Arguments of Nominalizations

4.1 Introduction

Until now we dealt with the task of identification and classification of arguments of verb predicates in a sentence. In order to be able to generate a sentence level semantic representation, it is necessary to be able to identify arguments of other possible predicates in a sentence like nominal predicates, adjectival predicates, prepositional predicates and predicates of auxiliary verbs. This chapter discusses the task of semantic role labeling as applied to a class of nominal predicates – nominalizations. A simple linguistic definition of nominalization is *the process of converting a verb into an abstract noun*. Take, for example, the following two sentence pairs, the second sentence in each pair is a nominalized version of the first. One important thing to note is that the verbs in the nominalized sentences are “make” and “took” respectively. A semantic analyzer that parses these sentences for the arguments of these verbs will miss the events – *complaining* and *walking*, that are central to understanding the meaning of the sentences, and which represented by the nominal predicates “complain” and “walk” respectively.

Of the vast literature on automatic semantic role labeling, very little deals with semantic role assigning properties of nouns. Most of what does, deals with nominalizations. However, given the lack of a corpus annotated with nominal predicates and their arguments, there have been no investigations into applying statistical algorithms that can automatically identify and label the arguments of nouns. To our knowledge, the

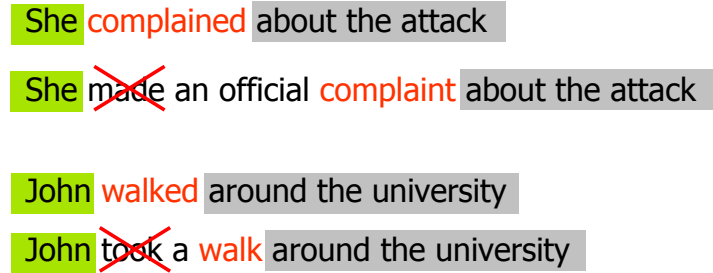


Figure 4.1: Nominal example.

only works that come closest are the rule-based system described by Hull and Gomez (1996) and the work of Lapata (2002) on interpreting the relation between the head of a nominalized compound and its modifier noun. With the availability of hand labeled argument information for nominal predicates through the FrameNet project, an investigation into the feasibility of automatically identifying nominal predicates using these data can be performed. We will look at two things: i) transforming features that were derived with respect to verbs, so that they will now be derived from nouns – most of which are quite straight forward, and unaffected. ii) how well do these transformed features identify the semantic properties of noun arguments. In other words, how good are these features at identifying and classifying nominal arguments? and are there any new set of features that are specific to nouns, in particular, nominalizations that we can add to the current set of features within ASSERT.

4.2 Semantic Annotation and Corpora

Among PropBank and FrameNet¹ (Baker et al., 1998) databases, only FrameNet contains argument annotations for nouns, therefore we use FrameNet for these experiments. For the purposes of this study, a human analyst went through the nominal predicates in FrameNet and selected those that were identified as nominalizations in

¹ <http://www.icsi.berkeley.edu/~framenet/>

NOMLEX (Macleod et al., 1998). Out of those, the analyst then selected ones that were eventive nominalizations.

These data comprise 7,333 annotated sentences, with 11,284 roles. There are 105 frames with about 190 distinct frame role types. A stratified sampling over predicates was performed to select 80% of these data for training, 10% for development and another 10% for testing.

4.3 Baseline System

We start with the ASSERT architecture as described before. Since the corpus that FrameNet was built using – the BNC, does not contain information on correct syntactic parses, we cannot replicate the first part of the verb argument study where we look at what the contribution of each feature was given perfect syntactic information. We generate the training data by using the Charniak parser (Charniak, 2000) to generate the syntactic parses. Another thing to note here is that there is a disparity in the corpus that the Charniak parser was trained on, which is the Wall Street Journal part of the Penn Treebank, and the corpus that we are using it to parse – the BNC corpus. Therefore it should be assumed that we will have more noise in the training data. During feature extraction for training, about 12% of the arguments did not match any constituent boundary owing to parse errors, which is not terribly bad as compared to the 6-10% deletions on the Brown corpus as mentioned in Table 6.3.

The baseline performance of this system is shown in Table 4.1.

Task	P (%)	R (%)	$F_{\beta=1}$	A (%)
Id.	81.7	65.7	72.8	
Classification	-	-	-	70.9
Id. + Classification	65.7	42.1	51.4	

Table 4.1: Baseline performance on all three tasks.

4.4 New Features

Following are some new features that were added to the system, along with an intuitive justification. Some of these features don't exist for some constituents. In those cases, the respective feature values are set to "UNK". Almost all the new features that we used for the verb predicates, except the CCG features, were added to the baseline system. We will re-list some of the features that we used for the verb predicates with an intuitive justification as to why we thought those features would be useful for the task of identifying arguments of nominalizations. For some features that do not have any special connotation in the nominal counterpart of the task, but which nevertheless added to the performance owing to some inherent generalization power for the overall task of identifying semantic arguments, will just be listed.

Frame – The frame instantiated by a certain sense of the predicate in a sentence.

This is an oracle feature.

Selected words/POS in constituent – Nominal predicates tend to assign arguments, most commonly through post-nominal of-complements, possessive phenomenal modifiers, etc. The values of the first and last word in the constituent were added as two separate features. Another two feature represented the part of speech of these words.

Ordinal constituent position – Arguments of nouns tend to be located closer to the predicate than those for verbs. This feature captures the ordinal position of the constituent to the left or right of the predicate on a left or right tree traversal, eg., first PP from the predicate, second NP from the predicate, etc. This feature along with the position will encode the before/after information for the constituent.

Constituent tree distance

Intervening verb features – Support verbs play an important role in realizing the arguments of nominal predicates. Three classes of intervening verbs were used: i) verbs of being – ones with part of speech AUX, ii) light verbs – a small set of known light verbs eg., make, take, have, etc., and iii) other verbs – with part of speech VBx. Three features were added for each: i) a binary feature indicating the presence of the verb in between the predicate and the constituent ii) the actual word as a feature, and iii) the path through the tree from the constituent to the verb. The following example could explain the intuition behind this feature:

1. [*Speaker* Leapor] *makes* general [*Predicate* assertions] [*Topic* about marriage]

Predicate NP expansion rule – This is the noun equivalent of the verb sub-categorization feature used by Gildea and Jurafsky (2002). This is the expansion rule instantiated by the syntactic parser, for the lowermost NP in the tree, encompassing the predicate. This would tend to cluster NPs with a similar internal structure and would thus help finding argumentive modifiers.

Use noun head for prepositional phrase constituents

Constituent sibling features

Partial-path from constituent to predicate

Is predicate plural – A binary feature indicating whether the predicate is singular or plural as they tend to have different argument selection properties.

Genitives in constituent – This is a binary feature which is true if there is a genitive word (one with the part of speech POS, PRP, PRP\$ or WP\$) in the constituent, as these tend to be subject/object markers for nominal arguments. The following example helps clarify this notion:

1. [*Speaker* Burma 's] [*Phenomenon* oil] [*Predicate* search] hits virgin forests

Constituent parent features

Verb dominating predicate – The head word of the first VP ancestor of the predicate.

Named Entities in Constituent

4.5 Best System Performance

Table 4.2 shows the improved performance numbers.

Task	P (%)	R (%)	$F_{\beta=1}$	A (%)
Id.	83.8	70.0	76.3	
Classification (w/o Frame)	-	-	-	73.1
Classification (with Frame)	-	-	-	80.9
Id. + Classification (one-pass, w/o Frame)	69.4	47.6	56.5	
Id. + Classification (two-pass, w/o Frame)	62.2	53.1	57.3	
Id. + Classification (two-pass, with Frame)	69.4	59.2	63.9	

Table 4.2: Best performance on all three tasks.

4.6 Feature Analysis

For the task of *argument identification*, features 2, 3, 4, 5 (the verb itself, path to light-verb and presence of a light verb), 6, 7, 9, 10 and 13 contributed positively to the performance. Interestingly the Frame feature degrades performance significantly. A new classifier was trained using all the features that contributed positively to the performance and the $F_{\beta=1}$ score increased from the baseline of 72.8% to 76.3% ($\chi^2; p < 0.05$).

For the task of *argument classification*, adding the Frame feature to the baseline features, provided the most significant improvement, increasing the classification accuracy from 70.9% to 79.0% ($\chi^2; p < 0.05$). All other features added one-by-one to the

baseline did not bring any significant improvement to the baseline. All the features together produced a classification accuracy of 80.9%. Since the Frame feature is an oracle, it would be interesting to find out what all the other features combined contributed. An experiment was run with all features, except Frame, added to the baseline, and this produced an accuracy of 73.1%, which is not a statistically significant improvement over the baseline of 70.9%.

For the task of *argument identification and classification*, features 8 and 11 (right sibling head word part of speech) hurt performance. A classifier was trained using all the features that contributed positively to the performance and the resulting system had an improved $F_{\beta=1}$ score of 56.5% compared to the baseline of 51.4% ($\chi^2; p < 0.05$).

A significant subset of features that contribute marginally to the classification performance, hurt the identification task. Therefore, it was decided to perform a two-step process in which the set of features that gave optimum performance for the argument identification task would be used and all likely argument nodes shall be identified. Then, for those nodes, all the available features will be used to classify them into one of the possible classes. This “two-pass” system performs slightly better than the “one-pass” mentioned earlier. Again, the second pass of classification was performed with and without the Frame feature.

4.7 Discussion

These preliminary results tend to indicate that the task of labeling arguments of nominalized predicates is more difficult than that of verb predicates. For a training set of 5,000 sentences annotated with verb arguments, the $F_{\beta=1}$ on a standard test set reported by Pradhan et al. (2003a) was 71.9%, as opposed to that of 57.3% using approximately 7,500 sentences training data, a super-set of the features, and a similar training algorithm. Although the comparison is not exact, it gives some insight.

Chapter 5

Different Syntactic Views

In this Chapter we will once again focus our attention to the tasks of identifying and classifying the arguments of verb predicates. After performing a detailed error analysis of our baseline system it was found that the identification problem poses a significant bottleneck to improving overall system performance. The baseline system's accuracy on the task of labeling nodes known to represent semantic arguments is 90%. On the other hand, the system's performance on the identification task is quite a bit lower, achieving only 80% recall with 86% precision. There are two sources of these identification errors: i) failures by the system to identify all and only those constituents that correspond to semantic roles, *when those constituents are present in the syntactic analysis*, and ii) failures by the syntactic analyzer to provide the constituents that align with correct arguments. The work we present in this chapter is tailored to address these two sources of error in the identification problem.

We then report on experiments that address the problem of arguments missing from a given syntactic analysis. We investigate ways to combine hypotheses generated from semantic role taggers trained using different syntactic views – one trained using the Charniak parser (Charniak, 2000), another on a rule-based dependency parser – Minipar (Lin, 1998b), and a third based on a flat, shallow syntactic chunk representation (Hacioglu, 2004a). We show that these three views complement each other to improve performance.

5.1 Baseline System

For these experiments, we use Feb 2004 release of PropBank.

The baseline feature set is a combination of features introduced by Gildea and Jurafsky (Gildea and Jurafsky, 2002) and ones proposed in Pradhan et al., (Pradhan et al., 2004) (also mentioned earlier), Surdeanu et al., (Surdeanu et al., 2003) and the *syntactic-frame* feature proposed in Xue and Palmer (2004). Table 5.1 lists the features used.

Table 5.2 shows the performance of the system using Treebank parses (**HAND**) and using parses produced by a Charniak parser (**AUTOMATIC**). Precision (**P**), Recall (**R**) and F_1 scores are given for the identification and combined tasks, and Classification Accuracy (**A**) for the classification task.

Classification performance using Charniak parses is about 3% absolute worse than when using Treebank parses. On the other hand, argument identification performance using Charniak parses is about 12.7% absolute worse. Half of these errors – about 7% are due to missing constituents, and the other half – about 6% are due to misclassifications.

Motivated by this severe degradation in argument identification performance for automatic parses, we examined a number of techniques for improving argument identification. We decided to combine parses from different syntactic representations.

PREDICATE LEMMA
PATH: Path from the constituent to the predicate in the parse tree.
POSITION: Whether the constituent is before or after the predicate.
VOICE
PREDICATE SUB-CATEGORIZATION
PREDICATE CLUSTER
HEAD WORD: Head word of the constituent.
HEAD WORD POS: POS of the head word
NAMED ENTITIES IN CONSTITUENTS: 7 named entities as 7 binary features.
PARTIAL PATH: Path from the constituent to the lowest common ancestor of the predicate and the constituent.
VERB SENSE INFORMATION: Oracle verb sense information from PropBank
HEAD WORD OF PP: Head of PP replaced by head word of NP inside it, and PP replaced by <i>PP-preposition</i>
FIRST AND LAST WORD/POS IN CONSTITUENT
ORDINAL CONSTITUENT POSITION
CONSTITUENT TREE DISTANCE
CONSTITUENT RELATIVE FEATURES: Nine features representing the phrase type, head word and head word part of speech of the parent, and left and right siblings of the constituent.
TEMPORAL CUE WORDS
DYNAMIC CLASS CONTEXT
SYNTACTIC FRAME
CONTENT WORD FEATURES: Content word, its POS and named entities in the content word

Table 5.1: Features used in the Baseline system

ALL ARGS	Task	P	R	F ₁	A
		(%)	(%)		(%)
HAND	Id.	96.2	95.8	96.0	
	Classification	-	-	-	93.0
	Id. + Classification	89.9	89.0	89.4	
AUTOMATIC	Id.	86.8	80.0	83.3	
	Classification	-	-	-	90.1
	Id. + Classification	80.9	76.8	78.8	

Table 5.2: Baseline system performance on all tasks using Treebank parses and automatic parses on PropBank data.

5.2 Alternative Syntactic Views

Adding new features can improve performance when the syntactic representation being used for classification contains the correct constituents. Additional features can't recover from the situation where the parse tree being used for classification doesn't contain the correct constituent representing an argument. Such parse errors account for about 7% absolute of the errors (or, about half of 12.7%) for the Charniak parse based system. Figure 5.1 shows how a wrong attachment decision in the parsing process could lead to the deletion of a node that represents an argument, and subsequently make it impossible for our current system architecture to recover from it. To address these errors, we added two additional parse representations: i) Minipar dependency parser, and ii) chunking semantic role labeler (Hacioglu et al., 2004). The hypothesis is that these parsers will produce different, and possibly complementary, errors since they represent different syntactic views. The Charniak parser is trained on the Penn Treebank corpus. Minipar is a rule based dependency parser. The chunking semantic role labeler is trained on PropBank and produces a flat syntactic representation that is very different from the full parse tree produced by Charniak. A combination of the three different parses could produce better results than any single one.

5.2.1 Minipar-based Semantic Labeler

Minipar (Lin, 1998b; Lin and Pantel, 2001) is a rule-based dependency parser. It outputs dependencies between a word called *head* and another called *modifier*. Each word can modify at most one word. The dependency relationships form a dependency tree.

The set of words under each node in Minipar's dependency tree form a contiguous segment in the original sentence and correspond to the constituent in a constituent tree. Figure 5.2 shows how the arguments of the predicate "kick" map to the nodes in a phrase

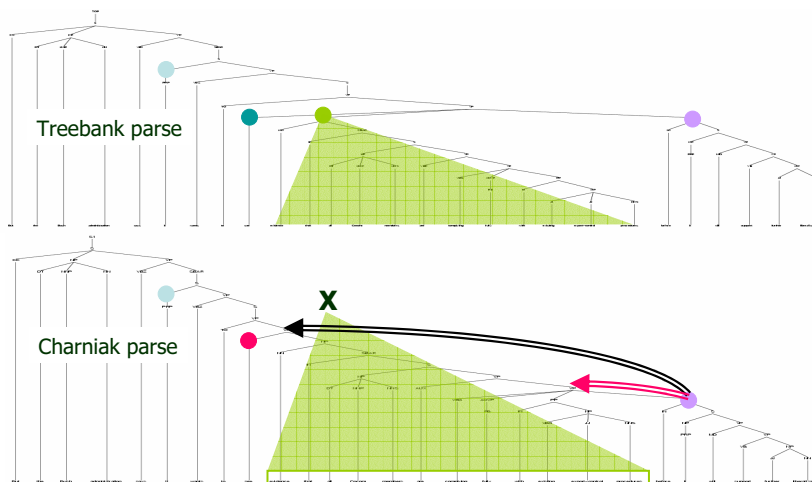


Figure 5.1: Illustration of how a parse error affects argument identification.

structure grammar tree as well as the nodes in a Minipar parse tree.

We formulate the semantic labeling problem in the same way as in a constituent structure parse, except we classify the nodes that represent head words of constituents. A similar formulation using dependency trees derived from Treebank was reported in Hacıoglu (Hacıoglu, 2004b). In that experiment, the dependency trees were derived from Treebank parses using head word rules. Here, an SVM is trained to assign PropBank argument labels to nodes in Minipar dependency trees using the following features:

Table 5.4 shows the performance of the Minipar-based semantic role labeler.

Minipar performance on the PropBank corpus is substantially worse than the Charniak based system. This is understandable from the fact that Minipar is not designed to produce constituents that would exactly match the constituent segmentation

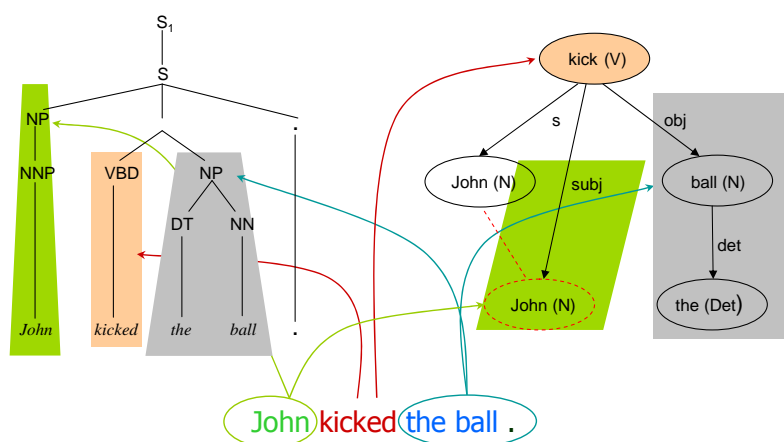


Figure 5.2: PSG and Minipar views.

used in Treebank. In the test set, about 37% of the arguments do not have corresponding constituents that match its boundaries. In experiments reported by Hacıoglu (Hacıoglu, 2004b), a mismatch of about 8% was introduced in the transformation from Treebank trees to dependency trees. Using an errorful automatically generated tree, a still higher mismatch would be expected. In case of the CCG parses, as reported by Gildea and Hockenmaier (2003), the mismatch was about 23%. A more realistic way to score the performance is to score tags assigned to head words of constituents, rather than considering the exact boundaries of the constituents as reported by Gildea and Hockenmaier (2003). The results for this system are shown in Table 5.5.

PREDICATE LEMMA
HEAD WORD: The word representing the node in the dependency tree.
HEAD WORD POS: Part of speech of the head word.
POS PATH: This is the path from the predicate to the head word through the dependency tree connecting the part of speech of each node in the tree.
DEPENDENCY PATH: Each word that is connected to the head word has a dependency relationship to the word. These are represented as labels on the arc between the words. This feature is the dependencies along the path that connects two words.
VOICE
POSITION

Table 5.3: Features used in the Baseline system using Minipar parses.

Task	P	R	F ₁
	(%)	(%)	
Id.	73.5	43.8	54.6
Id. + Classification	66.2	36.7	47.2

Table 5.4: Baseline system performance on all tasks using Minipar parses.

Task		P	R	F ₁
		(%)	(%)	
CHARNIAK	Id.	92.2	87.5	89.8
	Id. + Classification	85.9	81.6	83.7
MINIPAR	Id.	83.3	61.1	70.5
	Id. + Classification	72.9	53.5	61.7

Table 5.5: Head-word based performance using Charniak and Minipar parses.

5.2.2 Chunk-based Semantic Labeler

Hacioglu has previously described a chunk based semantic labeling method (Hacioglu et al., 2004). This system uses SVM classifiers to first chunk input text into flat chunks or base phrases, each labeled with a syntactic tag. A second SVM is trained to assign semantic labels to the chunks. Figure 5.3 shows a schematic of the chunking process.

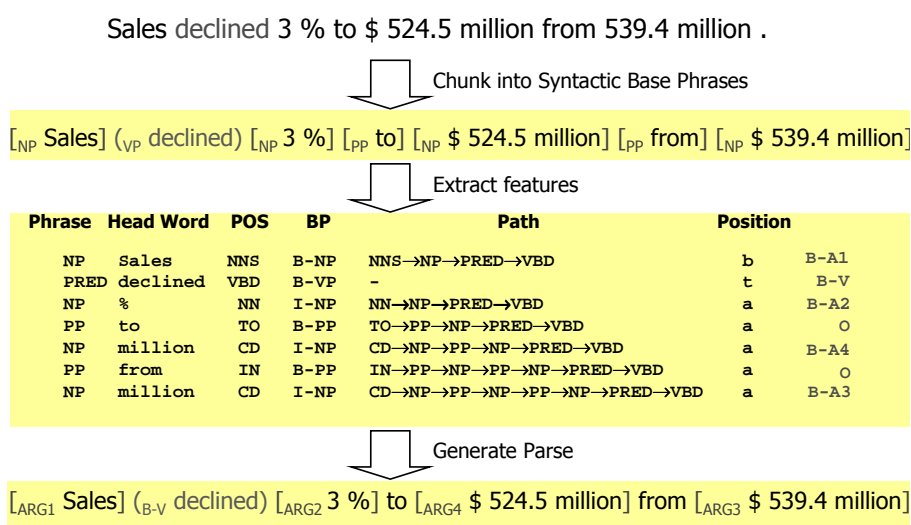


Figure 5.3: Semantic Chunker.

Table 5.6 lists the features used by this classifier. For each token (base phrase) to be tagged, a set of features is created from a fixed size context that surrounds each token. In addition to the above features, it also uses previous semantic tags that have already been assigned to the tokens contained in the linguistic context. A 5-token sliding window is used for the context.

SVMs were trained for begin (B) and inside (I) classes of all arguments and outside (O) class for a total of 78 one-vs-all classifiers. Again, TinySVM along with YamCha (Kudo and Matsumoto, 2000, 2001) are used as the SVM training and test software.

WORDS
PREDICATE LEMMAS
PART OF SPEECH TAGS
BP POSITIONS: The position of a token in a BP using the IOB2 representation (e.g. B-NP, I-NP, O, etc.)
CLAUSE TAGS: The tags that mark token positions in a sentence with respect to clauses.
NAMED ENTITIES: The IOB tags of named entities.
TOKEN POSITION: The position of the phrase with respect to the predicate. It has three values as "before", "after" and "-" (for the predicate)
PATH: It defines a flat path between the token and the predicate
CLAUSE BRACKET PATTERNS
CLAUSE POSITION: A binary feature that identifies whether the token is inside or outside the clause containing the predicate
HEADWORD SUFFIXES: suffixes of headwords of length 2, 3 and 4.
DISTANCE: Distance of the token from the predicate as a number of base phrases, and the distance as the number of VP chunks.
LENGTH: the number of words in a token.
PREDICATE POS TAG: the part of speech category of the predicate
PREDICATE FREQUENCY: Frequent or rare using a threshold of 3.
PREDICATE BP CONTEXT: The chain of BPs centered at the predicate within a window of size -2/+2.
PREDICATE POS CONTEXT: POS tags of words immediately preceding and following the predicate.
PREDICATE ARGUMENT FRAMES: Left and right core argument patterns around the predicate.
NUMBER OF PREDICATES: This is the number of predicates in the sentence.

Table 5.6: Features used by chunk based classifier.

	P	R	F_1
	(%)	(%)	
Id. and Classification	72.6	66.9	69.6

Table 5.7: Semantic chunker performance on the combined task of Id. and classification.

Table 5.7 shows the system performances on the PropBank test set for the chunk-based system.

5.3 Combining Semantic Labelers

We combined the semantic parses as follows: i) scores for arguments were converted to calibrated probabilities, and arguments with scores below a threshold value were deleted. Separate thresholds were used for each semantic role labeler. ii) For the remaining arguments, the more probable ones among overlapping ones were selected. In the chunked system, an argument could consist of a sequence of chunks. The probability assigned to the begin tag of an argument was used as the probability of the sequence of chunks forming an argument. Table 5.8 shows the performance improvement after the combination. Again, numbers in parentheses are respective baseline performances.

TASK	P	R	F ₁
	(%)	(%)	
Id.	85.9 (86.8)	88.3 (80.0)	87.1 (83.3)
Id. + Class.	81.3 (80.9)	80.7 (76.8)	81.0 (78.8)

Table 5.8: Constituent-based best system performance on argument identification and argument identification and classification tasks after combining all three semantic parses.

The main contribution of combining both the Minipar based and the Charniak-based semantic role labeler was significantly improved performance on ARG1 in addition to slight improvements to some other arguments. Table 5.9 shows the effect on selected arguments on sentences that were altered during the the combination of Charniak-based and Chunk-based parses.

A marked increase in number of propositions for which all the arguments were identified correctly from 0% to about 46% can be seen. Relatively few predicates, 107 out of 4500, were affected by this combination.

To give an idea of what the potential improvements of the combinations could

Number of Propositions	107
Percentage of perfect props before combination	0.00
Percentage of perfect props after combination	45.95

	Before			After		
	P	R	F ₁	P	R	F ₁
	(%)	(%)		(%)	(%)	
Overall	94.8	53.4	68.3	80.9	73.8	77.2
ARG0	96.0	85.7	90.5	92.5	89.2	90.9
ARG1	71.4	13.5	22.7	59.4	59.4	59.4
ARG2	100.0	20.0	33.3	50.0	20.0	28.5
ARGM-DIS	100.0	40.0	57.1	100.0	100.0	100.0

Table 5.9: Performance improvement on parses changed during pair-wise Charniak and Chunk combination.

be, we performed an oracle experiment for a combined system that tags head words instead of exact constituents as we did in case of Minipar-based and Charniak-based semantic role labeler earlier. In case of chunks, first word in prepositional base phrases was selected as the head word, and for all other chunks, the last word was selected to be the head word. If the correct argument was found present in either the Charniak, Minipar or Chunk hypotheses then that was selected. The results for this are shown in Table 5.10.

Table 5.11 shows the performance improvement in the actual system for pairwise combination of the semantic role labeler and one using all three.

Task		P	R	F ₁
		(%)	(%)	
C	Id.	92.2	87.5	89.8
	Id. + Classification	85.9	81.6	83.7
C+M	Id.	98.4	90.6	94.3
	Id. + Classification	93.1	86.0	89.4
C+CH	Id.	98.9	88.8	93.6
	Id. + Classification	92.5	83.3	87.7
C+M+CH	Id.	99.2	92.5	95.7
	Id. + Classification	94.6	88.4	91.5

Table 5.10: Performance improvement on head word based scoring after oracle combination. Charniak (C), Minipar (M) and Chunker (CH).

Task		P	R	F ₁
		(%)	(%)	
C	Id.	92.2	87.5	89.8
	Id. + Classification	85.9	81.6	83.7
C+M	Id.	91.7	89.9	90.8
	Id. + Classification	85.0	83.9	84.5
C+CH	Id.	91.5	91.1	91.3
	Id. + Classification	84.9	84.3	84.7
C+M+CH	Id.	91.5	91.9	91.7
	Id. + Classification	85.1	85.5	85.2

Table 5.11: Performance improvement on head word based scoring after combination. Charniak (C), Minipar (M) and Chunker (CH).

5.4 Improved Architecture

Some of the systems we discussed until now use features based on syntactic constituents produced by a syntactic parser (Pradhan et al., 2003b, 2004) and others use only a flat syntactic representation produced by a syntactic chunker (Hacioglu et al., 2003; Hacioglu and Ward, 2003; Hacioglu, 2004a; Hacioglu et al., 2004). The latter approach lacks the information provided by the hierarchical syntactic structure, and the former imposes a limitation that the possible candidate roles should be one of the nodes already present in the syntax tree. We found that, while the chunk based systems are very efficient and robust, the systems that use features based on full syntactic parses are generally more accurate. Analysis of the source of errors for the parse constituent based systems showed that incorrect parses were a major source of error. The syntactic parser did not produce any constituent that corresponded to the correct segmentation for the semantic argument. In Pradhan et al. (2005b), we reported on a first attempt to overcome this problem by combining semantic role labels produced from different syntactic parses. The hope is that the syntactic parsers will make different errors, and that combining their outputs will improve on either system alone. This initial attempt used features from a Charniak parser, a Minipar parser and a chunk based parser. It did show some improvement from the combination, but the method for combining the information was heuristic and sub-optimal. Now, we propose what we believe is an improved framework for combining information from different syntactic views. Our goal is to preserve the robustness and flexibility of the segmentation of the phrase-based chunker, but to take advantage of features from full syntactic parses. We also want to combine features from different syntactic parses to gain additional robustness. To this end, we use features generated from a Charniak parser and a Collins parser, as supplied for the CoNLL-2005 closed task.

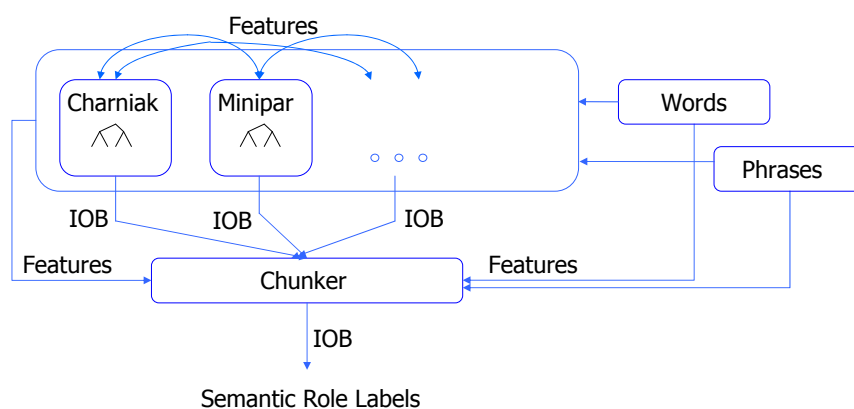


Figure 5.4: New Architecture.

5.5 System Description

The general framework is to train separate semantic role labeling systems for each of the parse tree views, and then to use the role arguments output by these systems as additional features in a semantic role classifier using a flat syntactic view. The constituent based classifiers walk a syntactic parse tree and classify each node as NULL (no role) or as one of the set of semantic roles. Chunk based systems classify each base phrase as being the B(eginning) of a semantic role, I(nside) a semantic role, or O(utside) any semantic role (ie. NULL). This is referred to as an IOB representation (Ramshaw and Marcus, 1995). The constituent level roles are mapped to the IOB representation used by the chunker. The IOB tags are then used as features for a separate base-phase semantic role labeler (chunker), in addition to the standard set of features used by the chunker. An n-fold cross-validation paradigm is used to train the constituent based role classifiers and the chunk based classifier.

For the system reported here, two full syntactic parsers were used, a Charniak parser and a Collins parser. Features were extracted by first generating the Collins and

Charniak syntax trees from the word-by-word decomposed trees in the CoNLL data. The chunking system for combining all features was trained using a 4-fold paradigm. In each fold, separate SVM classifiers were trained for the Collins and Charniak parses using 75% of the training data. That is, one system assigned role labels to the nodes in Charniak based trees and a separate system assigned roles to nodes in Collins based trees. The other 25% of the training data was then labeled by each of the systems. Iterating this process 4 times created the training set for the chunker. After the chunker was trained, the Charniak and Collins based semantic labelers were then retrained using all of the training data.

Two pieces of the system have problems scaling to large training sets – the final chunk based classifier and the NULL *vs* NON-NULL classifier for the parse tree syntactic views. Two techniques were used to reduce the amount of training data – active sampling and NULL filtering. The active sampling process was performed as follows. We first train a system using 10k seed examples from the training set. We then labeled an additional block of data using this system. Any sentences containing an error were added to the seed training set. The system was retrained and the procedure repeated until there were no misclassified sentences remaining in the training data. The set of examples produced by this procedure was used to train the final NULL *vs* NON-NULL classifier. The same procedure was carried out for the chunking system. After both these were trained, we tagged the training data using them and removed all most likely NULLs from the data.

In addition to the features extracted from the parse tree being labeled, five features were extracted from the other parse tree (phrase, head word, head word POS, path and predicate sub-categorization). So for example, when assigning labels to constituents in a Charniak parse, all of the features in Table 1 were extracted from the Charniak tree, and in addition phrase, head word, head word POS, path and sub-categorization were extracted from the Collins tree. We have previously determined that using different sets of features for each argument (role) achieves better results than using the same set of

features for all argument classes. A simple feature selection was implemented by adding features one by one to an initial set of features and selecting those that contribute significantly to the performance. As described in Pradhan et al. (2004), we post-process lattices of n-best decision using a trigram language model of argument sequences.

SVMs were trained for begin (B) and inside (I) classes of all arguments and an outside (O) class. One particular advantage of this architecture, as depicted in Figure 5.5 is that the final segmentation does not have to necessarily be adhering to one of the input segmentations, and depending on the provided information in terms of features, the classifier can generate a new, better segmentation.

Words	View-1	View-2	Reference	Hypothesis
The	B-A1	O	B-A1	B-A1
slickly	I-A1	O	I-A1	I-A1
produced	I-A1	O	I-A1	I-A1
series	I-A1	O	I-A1	I-A1
has	O	O	O	O
been	O	O	O	O
criticized	B-V	B-V	B-V	B-V
by	B-A0	B-A0	B-A0	B-A0
London	I-A0	I-A0	I-A0	I-A0
's	I-A0	I-A0	I-A0	I-A0
financial	I-A0	I-A0	I-A0	I-A0
cognoscenti	I-A0	I-A0	I-A0	I-A0
as	B-A2	B-A2	B-A2	B-A2
inaccurate	I-A2	I-A2	I-A2	I-A2
in	B-AM-MNR	B-AM-MNR	I-A2	I-A2
detail	I-AM-MNR	I-AM-MNR	I-A2	I-A2
,	O	O	O	O
but	O	O	O	O
.				
.				

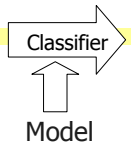


Figure 5.5: Example classification using the new architecture.

5.6 Results

We participated in the CoNLL shared task on SRL (Carreras and Màrquez, 2005). This gave us an opportunity to test our integrated architecture. For this evaluation, various syntactic features were provided, including output of a Base Phrase chunker, Charniak parser, Collins' parser, Clause tagger and Named Entities. In this evaluation, in addition to the WSJ section-23, a portion of a completely different genre of text

from The Brown Corpus Kučera and Francis (1967) was used. The section of the Brown Corpus that was used for evaluation contains prose from “general fiction” and represents quite a different genre of material from what the SRL systems were trained on. This was done to investigate into the robustness of these systems.

Table 5.12 shows the performance of the new architecture on the CoNLL 2005 test set. As it can be seen, almost all the state-of-the-art systems – including the one described so far, suffered a 10 point drop in the F measure.

	P	R	F
WSJ	82.95%	74.75%	78.63
Brown	74.49%	63.30%	68.44
WSJ+Brown	81.87%	73.21%	77.30

Table 5.12: Performance of the integrated architecture on the CoNLL-2005 shared task on semantic role labeling.

Chapter 6

Robustness Experiments

So far most of the recent work on SRL systems has been focused on improving the labeling performance on a test set belonging to the same genre of text as the training set. Both, the Treebank on which the syntactic parser is trained, and the PropBank on which the SRL systems are trained represent articles from the year 1989 of the Wall Street Journal. Part of the reason for this is the lack of data tagged with similar semantic argument structure in multiple genres of text. At this juncture it is quite possible that these tools are being subject to the effects of over-training to this genre of text, and instead of any improvements to the system reflecting further progress in the field, are getting tuned to this style of journalistic text. It is also important for this technology to be widely accepted that it performs reasonably well on text that does not represent the Wall Street Journal.

As a pilot experiment, Pradhan et al. (2004, 2005a) reported some preliminary analysis on a small test set that was tagged for a small portion of about 400 sentences from the AQUAINT corpus, which is a collection of articles from the New York Times, Xinhua, and FBIS. Although this collection also represents newswire text, it was aimed at finding out whether an incremental change within the same general domain that occurred in a different time period and presumably representing different entities and events and maybe a different style would impact the performance of the SRL system. As a matter of fact, the system performance dropped from F-scores in the high 70s to

that in the low 60s. As we now know, this is not much different from the performance obtained on a test set from the Brown corpus which represents quite different style of text. Fortunately, Palmer et al. (2005b) have also recently PropBanked a significant portion of the Brown corpus, and therefore it is possible to perform a more systematic analysis of the portability of SRL systems from one genre of text to another.

6.1 The Brown Corpus

The Brown corpus is a Standard Corpus of American English that consists of about one million words of English text printed in the calendar year 1961 (Kučera and Francis, 1967). The corpus contains about 500 samples of 2000+ words each. The idea behind creating this corpus was to create a heterogeneous sample of English text so that it would be useful for comparative language studies. It is comprised of the following sections:

- A. Press Reportage
- B. Press Editorial
- C. Press Reviews (theater, books, music and dance)
- D. Religion
- E. Skills and Hobbies
- F. Popular Lore
- G. Belles Lettres, Biography, Memoirs, etc.
- H. Miscellaneous
- J. Learned
- K. General Fiction

L. Mystery and Detective Fiction

M. Science Fiction

N. Adventure and Western Fiction

P. Romance and Love Story

R. Humor

6.2 Semantic Annotation

The Release 3 of the Penn Treebank contains the hand parsed syntactic trees of a subset of the Brown Corpus – sections F, G, K, L, M, N, P and R. Sections belonging to the newswire genre were especially not considered for Treebanking because a considerable amount of the similar material was already available as the WSJ portion of the Treebank. Palmer et al. (2005b) have recently PropBanked a significant portion of this Treebanked Brown corpus. The PropBanking philosophy is the same as described earlier. In all, about 17,500 predicates are tagged with their semantic arguments. For these experiments we use a limited release of PropBank dated September 2005.

Table 6.1 gives the amount of predicates that have been tagged from each section:

Section	Total Predicates	Total Lemmas
F	926	321
G	777	302
K	8231	1476
L	5546	1118
M	167	107
N	863	269
P	788	252
R	224	140

Table 6.1: Number of predicates that have been tagged in the PropBanked portion of Brown corpus

We used the tagging scheme used in the CoNLL shared task to generate the

training and test data. All the scoring in the following experiments was done using the scoring scripts provided for the CoNLL 2005 shared task. The version of the Brown corpus that we used for our experiments did not have frame sense information, so we decided not to use that as a feature.

6.3 Experiments

This section focuses on various experiments that we performed on the PropBanked Brown corpus and which could go some way in analyzing the factors that affect the portability of SRL systems, and might throw some light on what steps need to be taken to improving the same. In order to avoid confounding the effects of the two distinct views that we saw earlier – the top-down syntactic view that extracts features from a syntactic parse, and the bottom-up phrase chunking view – for all these experiments, except one, we will use the system that is based on classifying constituents in a syntactic tree with PropBank arguments.

6.3.1 Experiment 1: How does ASSERT trained on WSJ perform on Brown?

In this section we will more thoroughly analyze what happens when a SRL system is trained on semantic arguments tagged on one genre of text – the Wall Street Journal, and is used to label those in a completely different genre – the Brown corpus.

Part of the test set that was used for the CoNLL 2005 shared task comprised of 800 predicates from the section CK of Brown corpus. This is about 5% of the available PropBanked Brown predicates, so we decided to use the entire Brown corpus as a test set for this experiment and use ASSERT trained on WSJ sections 02-21 to tag its arguments.

6.3.1.1 Results

Table 6.2 gives the details of the performance over each of the eight different text genres. It can be seen that on an average, the F-score on the combined task of identification and classification is comparable to the ones obtained on the AQUAINT test set. It is interesting to note that although AQUAINT is a different text source, it is still essentially newswire text. However, even though Brown corpus has much more variety, on an average, the degradation in performance is almost identical. This tells us that maybe the models are tuned to the particular vocabulary and sense structure associated with the training data. Also, since the syntactic parser that is used for generating the parse trees is also heavily lexicalized, it could also have some impact on the accuracy of the parses, and the features extracted from them.

Train	Test	Id. F	Id. + Class F
PropBank	PropBank (WSJ)	87.4	81.2
PropBank	Brown (Popular lore)	78.7	65.1
PropBank	Brown (Biography, Memoirs)	79.7	63.3
PropBank	Brown (General fiction)	81.3	66.1
PropBank	Brown (Detective fiction)	84.7	69.1
PropBank	Brown (Science fiction)	85.2	67.5
PropBank	Brown (Adventure)	84.2	67.5
PropBank	Brown (Romance and love Story)	83.3	66.2
PropBank	Brown (Humor)	80.6	65.0
PropBank	Brown (All)	82.4	65.1

Table 6.2: Performance on the entire PropBanked Brown corpus.

In order to check the extent of the deletion errors owing to the parser mistakes which result in the constituents representing a valid node getting deleted, we generated the appropriate numbers which are shown in Table 6.3. These numbers are for top one parse.

It can be seen that, as expected, the parser deletes very few argument bearing nodes in the tree when it is trained and tested on the same corpus. However, this number

	Total	Misses	%
PropBank	12000	800	6.7
Brown (Popular lore)	2280	219	9.6
Brown (Biography, Memoirs)	2180	209	9.6
Brown (General fiction)	21611	1770	8.2
Brown (Detective fiction)	14740	1105	7.5
Brown (Science fiction)	405	23	5.7
Brown (Adventure)	2144	169	7.9
Brown (Romance and love Story)	1928	136	7.1
Brown (Humor)	592	61	10.3
Brown (All)	45880	3692	8.1

Table 6.3: Constituent deletions in WSJ test set and the entire PropBanked Brown corpus.

does not drastically degrade when text from quite a disparate collection is parsed. In the worst case, the error rate increases by about a factor of 1.5 ($10.3/6.7$) which goes some ways in explaining the reduction in the overall performance. This seems to indicate that the syntactic parser does not contribute heavily to the performance drop across genre.

6.3.2 Experiment 2: How well do the features transfer to a different genre?

Several researchers have come up with novel features that improve the performance of SRL systems on WSJ test set, but a question lingers as to whether the same features when used to train SRL systems on a different genre of text would contribute equally well? There are actually two facets to this issue. One is whether the features themselves – regardless of what text they are generated from, are useful as they seem to be, and another is whether the values of some features for a particular corpus tend to represent an idiosyncrasy of that corpus, and therefore artificially get weighted heavily. This experiment is designed to throw some light on this issue.

In this experiment, we wanted to remove the effect of errors in estimating the syntactic structure. Therefore, we used correct syntactic trees from the Treebank. We trained ASSERT on a Brown training set and tested it on a test set also from the Brown corpus. Instead of using the CoNLL 2005 test set which represents part of section CK,

we decided to use a stratified test set as used by the syntactic parsing community (Gildea, 2001). The test set is generated by selecting every 10th sentence in the Brown Corpus. We also held out a development set used by Bacchiani et al. (2006) to tune system parameters in the future. We did not perform any parameter tuning specially for this or any of the following experiments, and used the same parameters as that reported for the best performing version of ASSERT as reported in Table 3.19 of this thesis. We compare the performance on this test set with that obtained when ASSERT is trained using WSJ sections 00-21 and use section 23 for testing. For a more balanced comparison, we also retrained ASSERT on the same amount of data as used for training it on Brown, and tested it on section 23. As usual, trace information, and function tag information from the Treebank is stripped out.

6.3.2.1 Results

Table 6.4 shows that there is a very negligible difference in argument identification performance when ASSERT is trained on 14,000 predicates and 104,000 predicates from the WSJ. We can notice a considerable drop in classification accuracy though. Further, when ASSERT is trained on Brown training data and tested on the Brown test data, the argument identification performance is quite similar to the one that is obtained on the WSJ test set using ASSERT trained on Treebank WSJ parses. It tells us that the drop in argument classification accuracy is much more severe. We know that the predicate whose arguments are being identified, and the head word of the syntactic constituent being classified are both important features in the task of argument classification. This evidence tends to indicate one of the following: i) maybe the task of classification needs much more data to train, and that this is merely an effect of the quantity of data, ii) maybe the predicates and head words (or, words in general) in a homogeneous corpus such as the WSJ are used more consistently, and that the style is simple and therefore it becomes an easier task for classification as opposed to the various usages and senses

in a heterogeneous collection such as the Brown corpus, iii) the features that are used for classification are more appropriate for WSJ than for Brown.

6.3.3 Experiment 3: How much does correct structure help?

In this experiment we will try to analyze how well do the structural features – the ones such as path whose accuracy depends directly on the quality of the syntax tree, transfer from one genre to another.

SRL Train	SRL Test	Task	P (%)	R (%)	F	A (%)
WSJ (104k)	WSJ (5k)	Id.	97.5	96.1	96.8	
		Class.				93.0
		Id. + Class.	91.8	90.5	91.2	
WSJ (14k)	WSJ (5k)	Id.	96.3	94.4	95.3	
		Class.				86.1
		Id. + Class.	84.4	79.8	82.0	
Brown (14k)	Brown (1.6k)	Id.	95.7	94.9	95.2	
		Class.				80.1
		Id. + Class.	79.9	77.0	78.4	
WSJ (14k)	Brown (1.6k)	Id.	94.2	91.4	92.7	
		Class.				72.0
		Id. + Class.	71.8	65.8	68.6	

Table 6.4: Performance when ASSERT is trained using correct Treebank parses, and is used to classify test set from either the same genre or another. For each dataset, the number of examples used for training are shown in parenthesis

For this experiment we train ASSERT on PropBanked WSJ, using correct syntactic parses from the Treebank, and using that model to test the same Brown test set, also generated using correct Treebank parses.

6.3.3.1 Results

Table 6.4 shows that the syntactic information from WSJ transfers quite well to the Brown corpus. Once again we see, that there is a very slight drop in argument identification performance, but an even greater drop in the argument classification accuracy.

6.3.4 Experiment 4: How sensitive is semantic argument prediction to the syntactic correctness across genre?

Now that we know that if you have correct syntactic information, that it transfers well across genre for the task of identification, we would now like to find out what happens when you use errorful automatically generated syntactic parses.

For this experiment, we used the same amount of training data from WSJ as available in the Brown training set – that is about 14,000 predicates. The examples from WSJ were selected randomly. The Brown test set is the same as used in the previous experiment, and the WSJ test set is the entire section 23.

Recently there have been some improvements to the Charniak parser, and that provides us with an opportunity to experiment with its latest version that does n -best re-ranking as reported in Charniak and Johnson (2005) and one that uses self-training and re-ranking using data from the North American News corpus (NANC) and adapts much better to the Brown corpus (McClosky et al., 2006b,a). We also use another one that is trained on Brown corpus itself. The performance of these parsers as reported in the respective literature are shown in Table 6.5

Train	Test	F
WSJ	WSJ	91.0
WSJ	Brown	85.2
Brown	Brown	88.4
WSJ+NANC	Brown	87.9

Table 6.5: Performance of different versions of Charniak parser used in the experiments.

We describe the results of the following five experiments:

- (1) ASSERT is trained on features extracted from automatically generated parses of the PropBanked WSJ sentences. The syntactic parser – Charniak parser – is itself trained on the WSJ training sections of the Treebank. This is used to classify the section-23 of WSJ.

- (2) ASSERT is trained on features extracted from automatically generated parses of the PropBanked WSJ sentences. The syntactic parser – Charniak parser – is itself trained on the WSJ training sections of the Treebank. This is used to classify the Brown test set.
- (3) ASSERT is trained on features extracted from automatically generated parses of the PropBanked Brown corpus sentences. The syntactic parser is trained using the WSJ portion of the Treebank. This is used to classify the Brown test set.
- (4) ASSERT is trained on features extracted from automatically generated parses of the PropBanked Brown corpus sentences. The syntactic parser is trained using the Brown training portion of the Treebank. This is used to classify the Brown test set.
- (5) ASSERT is trained on features extracted from automatically generated parses of the PropBanked Brown corpus sentences. The syntactic parser is the version that is self-trained using 2,500,000 sentences from NANC, and where the starting version is trained only on WSJ data (McClosky et al., 2006a). This is used to classify the Brown test set.

6.3.4.1 Results

Table 6.6 shows the results of these experiments. For simplicity of discussion we have tagged the five setups as A., B., C., D., and E. Looking at setups B. and C. it can be seen that when the features used to train ASSERT are extracted using a syntactic parser that is trained on WSJ it performs at almost the same level on the task of identification, regardless of whether it is trained on the PropBanked Brown corpus or the PropBanked WSJ corpus. This, however, is about 5-6 F-score points lower than when all the three – the syntactic parser training set, ASSERT training set, and ASSERT test set, are

from the same genre – WSJ or Brown, as seen in A. and D. In case of the combined task the gap between the performance for set up B. and C. is about 10 points F-score apart (59.1 vs 69.8) Looking at the argument classification accuracies, we see that using a ASSERT trained on WSJ to test Brown sentences give a 12 point drop in F-score. Using ASSERT trained on Brown using WSJ trained syntactic parser seems to drop in accuracy by about 5 F-score points. When ASSERT is trained on Brown using syntactic parser also trained on Brown, we get a quite similar classification performance, which is again about 5 points lower than what we get using all WSJ data. This shows lexical semantic features might be very important to get a better argument classification on Brown corpus.

Setup	Parser Train	SRL Train	SRL Test	Task	P (%)	R (%)	F	A (%)
A.	WSJ (40k – sec:00-21)	WSJ (14k)	WSJ (5k)	Id.	87.3	84.8	86.0	
				Class. Id. + Class.	77.5	69.7	73.4	84.1
B.	WSJ (40k – sec:00-21)	WSJ (14k)	Brown (1.6k)	Id.	81.7	78.3	79.9	
				Class.				72.1
				Id. + Class.	63.7	55.1	59.1	
C.	WSJ (40k – sec:00-21)	Brown (14k)	Brown (1.6k)	Id.	81.7	78.3	80.0	
				Class.				79.2
				Id. + Class.	78.2	63.2	69.8	
D.	Brown (20k)	Brown (14k)	Brown (1.6k)	Id.	87.6	82.3	84.8	
				Class.				78.9
				Id. + Class.	77.4	62.1	68.9	
E.	WSJ+NANC (2,500k)	Brown (14k)	Brown (1.6k)	Id.	87.7	82.5	85.0	
				Class.				79.9
				Id. + Class.	77.2	64.4	70.0	

Table 6.6: Performance on WSJ and Brown test set when ASSERT is trained on features extracted from automatically generated syntactic parses

6.3.5 Experiment 5: How much does combining syntactic views help overcome the errors?

At this point there seems to be quite a bit convincing evidence that the classification and not identification task, undergoes more degradation when going from one genre to another. What one would still like to see is how much does the integrated approach using both top-down syntactic information and bottom up chunk information buy us in moving from one genre to the other.

For this experiment we used the Syntactic parser trained on WSJ and one that is adapted through self-training using the NANC, and a base phrase chunker that is trained on WSJ Treebank, and use the integrated architecture as described in Section 5.4.

6.3.5.1 Results

As expected, we see a very small improvement in performance on the combined task of identification and classification. As the main contribution of this approach is to overcome the argument deletions, the improvement in performance is almost entirely owing to that.

Parser Train	BP Chunker Train	SRL Train	P (%)	R (%)	F
WSJ	WSJ	Brown	76.1	65.3	70.2
WSJ+NANC	WSJ	Brown	77.7	66.0	71.3

Table 6.7: Performance of the task of argument identification and classification using architecture that combines top down syntactic parses with flat syntactic chunks.

6.3.6 Experiment 6: How much data do we need to adapt to a new genre?

In general, it would be nice to know how much data from a new genre do we need to annotate and add to the training data of an existing labeler so that it can adapt itself to it and give the same level of performance when it is trained on that genre.

Fortunately, one section of the Brown corpus – section CK has about 8,200 predicates annotated. Therefore, we will take six different scenarios – two in which we will use correct Treebank parses, and the four others in which we will use automatically generated parses using the variations used before. All training sets start with the same number of examples as that of the Brown training set. We also happen to have a part of this section used as a test set for the CoNLL 2005 shared task. Therefore, we will use this as the test set for these experiments.

6.3.6.1 Results

Table 6.8 shows the result of these experiments. It can be seen that in all the six settings, the performance on the task of identification and classification improves gradually until about 5625 examples of section CK which is about 75% of the total added, above which it adds very little. It is very nice to note that even when the syntactic parser is trained on WSJ and the SRL is trained on WSJ, that adding 7,500 instances of this new genres allows it to achieve almost the same amount of performance as that achieved when all the three are from the same genre (67.2 vs 69.9) As for the task of argument identification, the incremental addition of data from the new genre shows only minimally improvement. The system that uses self-trained syntactic parser seems to perform slightly better than the rest of the versions that use automatically generated syntactic parses. Another point that might be worth noting is that the improvement on the identification performance is almost exclusively to the recall. The precision number are almost unaffected – except when the labeler is trained on WSJ PropBank data.

Parser	SRL	Id.			Id. + Class		
		P (%)	R (%)	F (%)	P (%)	R (%)	F (%)
Train	Train						
WSJ (Treebank parses)	WSJ (14k) (Treebank parses)						
	+0 examples from CK	96.2	91.9	94.0	74.1	66.5	70.1
	+1875 examples from CK	96.1	92.9	94.5	77.6	71.3	74.3
	+3750 examples from CK	96.3	94.2	95.1	79.1	74.1	76.5
	+5625 examples from CK	96.4	94.8	95.6	80.4	76.1	78.1
Brown (Treebank parses)	Brown (14k) (Treebank parses)						
	+0 examples from CK	96.1	94.2	95.1	77.1	73.0	75.0
	+1875 examples from CK	96.1	95.4	95.7	78.8	75.1	76.9
	+3750 examples from CK	96.3	94.6	95.3	80.4	76.9	78.6
	+5625 examples from CK	96.2	94.8	95.5	80.4	77.2	78.7
WSJ (40k)	WSJ (14k)						
	+0 examples from CK	83.1	78.8	80.9	65.2	55.7	60.1
	+1875 examples from CK	83.4	79.3	81.3	68.9	57.5	62.7
	+3750 examples from CK	83.9	79.1	81.4	71.8	59.3	64.9
	+5625 examples from CK	84.5	79.5	81.9	74.3	61.3	67.2
WSJ (40k)	Brown (14k)						
	+0 examples from CK	85.7	77.2	81.2	74.4	57.0	64.5
	+1875 examples from CK	85.7	77.6	81.4	75.1	58.7	65.9
	+3750 examples from CK	85.6	78.1	81.7	76.1	59.6	66.9
	+5625 examples from CK	85.7	78.5	81.9	76.9	60.5	67.7
Brown (20k)	Brown (14k)						
	+0 examples from CK	87.6	80.6	83.9	76.0	59.2	66.5
	+1875 examples from CK	87.4	81.2	84.1	76.1	60.0	67.1
	+3750 examples from CK	87.5	81.6	84.4	77.7	62.4	69.2
	+5625 examples from CK	87.5	82.0	84.6	78.2	63.5	70.1
WSJ+NANC (2,500k)	Brown (14k)						
	+0 examples from CK	89.1	81.7	85.2	74.4	60.1	66.5
	+1875 examples from CK	88.6	82.2	85.2	76.2	62.3	68.5
	+3750 examples from CK	88.3	82.6	85.3	76.8	63.6	69.6
	+5625 examples from CK	88.3	82.4	85.2	77.7	63.8	70.0
	+7500 examples from CK	88.9	82.9	85.8	78.2	64.9	70.9

Table 6.8: Effect of incrementally adding data from a new genre

Chapter 7

Conclusions and Future Work

7.1 Summary of Experiments

In this thesis, we have examined the problem of semantic role labeling through a series of experiments designed to show what features are useful for the task and how such features may be combined. A baseline system was developed and evaluated that represented the state-of-the-art at that time. New features were then evaluated in the context of the system, and the system was optimized for feature combinations. A novel method for combining various representations of syntactic information was introduced and evaluated. The system was extended to nominal predicates. Experiments were performed to evaluate the robustness of the system to genres of text other than the one it was trained on.

7.1.1 Performance Using Correct Syntactic Parses

We began with the set of features that were introduced by Gildea and Jurafsky, but used Support Vector Machine classifiers that were shown to perform better than their original formulation. The general approach of extracting features that are then used by SVM classifiers is followed through all of our experiments. After establishing a baseline performance with the G&J features, we investigated the effect on performance of adding many new features. In this process, feature salience experiments were conducted to determine the contribution of each of the individual features used. This

analysis showed that the path feature was particularly salient for the argument identification task, ie., identifying those nodes in the syntax tree that are associated with semantic arguments. While very useful, this feature does not generalize well because it is represented by very specific patterns. Creating more general versions of this feature did not help, but adding features that capture tree context information was very useful. An additional insight gained from analyzing the path feature was that, while it is very salient for the Identification task, it is not very useful for the classification task (classifying the role given that the constituent is an argument). Our architecture for the task uses a set of independent classifiers, one for each argument (including a NULL argument). The set of features optimal for one classifier is not necessarily optimal for the others. Therefore, we executed a feature selection procedure for each classifier to produce a more optimal set of features for each. Since the system used independent classifiers based on different subsets of features, each classifier output was calibrated to produce more accurate probabilities, so the outputs would be more comparable. Given correct syntactic parses as input, this system produced semantic role labels with an accuracy approaching human inter-annotator agreement accuracy which is around 90% (Palmer et al., 2005a).

7.1.2 Using Output from a Syntactic Parser

In most real applications, the system will not have access to human generated syntactic information, but must use output from a syntactic parser. We therefore measured performance using output from a state-of-the-art syntactic parser (Charniak, 2000) and compared it to performance of the system given human corrected syntactic parses. Use of the Charniak parser, which is reported to have an F-score of about 88 on the same test set using the parse-eval metric, resulted in a 10 point reduction in F-score.

7.1.3 Combining Syntactic Views

An analysis of errors from using syntactic parser output showed that a majority of the errors resulted from the case where there was no node in the parse tree that aligned with the correct argument. Since the labeling algorithm walked the parse trees classifying nodes, it could not produce correct output if there was no node in the tree corresponding to the argument. Our solution to this problem was to not rely on any one syntactic representation. We felt that using several different syntactic representations would be more robust than counting on any one parse to have all argument nodes represented. If different views tended to make different errors, then they could be combined to give complementary information. We chose three syntactic representations that we believed would provide complementary information:

Charniak - Statistical PSG parser trained on WSJ Treebank parses

Minipar - Rule based dependency parser not developed on WSJ data

Chunk parser - Trained on WSJ data but produces a flat structure

An oracle experiment showed that the three did make different errors and have potentially complementary information. The issue then was how to combine the three representations into a final decision. One obvious possibility is to train three separate systems for semantic role labeling, one for each of the syntactic representations. The argument classifications produced by each can then be assigned confidence scores and combined into a single lattice of arguments. A dynamic programming procedure can then be used to search the lattice to produce the argument sequence for each predicate that has the highest combined confidence. This was the first method that we evaluated for combining syntactic representations and it did provide slightly better performance than using any single view. However, this method has the disadvantage that the system must pick between the segmentations produced by each individual syntactic view and

cannot use the combined information to produce a new segmentation. To address this issue we developed a new architecture based on the chunking system. In this approach, semantic role labels with confidence scores are still produced separately for each syntactic view. However, when combining this information, the semantic roles are used as features, along with features from the original classifiers, to input to an SVM based chunking system. The chunker uses all of the features to produce a new segmentation and labeling, which is the final output. This system is able to produce a segmentation and labeling, using all of the information, that is different from those produced by any of the original classifiers. This new architecture was shown to give an additional performance gain over the original combination method.

7.2 What does it mean to be correct?

Another issue that arose when looking at performance of different syntactic views was the question of how the role labels were scored in evaluation. Labels were scored correct only if they matched the PropBank annotation exactly. Both the bracketing and the label had to match. Since PropBank and the Charniak parser were both developed on the Penn Treebank corpus, and were based on the same syntactic structures, it would be expected to match the PropBank labeling better than the other representations. But does a better score here imply that the output is more usable for any applications that would build on the role labels? It may often be the case that the specific bracketing is not really important, but the critical information is the relation of the argument headword to the predicate. Scoring the output of the algorithm using this strategy gave a much higher performance with an F-score of about 85 (Table 5.11).

7.3 Robustness to Genre of Data

Both the Charniak parser and the PropBank corpus were developed using the Wall Street Journal corpus (WSJ articles from the late 1980s), and are therefore sub-

ject to effects of over-training to this specific genre of data. In order to determine the robustness of the system to a change in genre of the data, we ran the system on test sets drawn from two other sources of text, the AQUAINT corpus and the Brown corpus. The AQUAINT corpus contains a collection of news articles from AP, NYT 1996 to 2000. The Brown corpus on the other hand, is a corpus of Standard American English compiled by Kučera and Francis (1967) It contains about a million words from about 15 different text categories, including press reportage, editorials, popular lore, science fiction, etc. The Semantic Role Labeling (Classification + Identification) F-score dropped from 81.2 for the PropBank test set to 62.8 for AQUAINT data and 65.1 for Brown data. Even though the AQUAINT data is newswire text, there is still a significant drop in performance. In general, these results point to over-training to the WSJ data. Analysis showed that errors in the syntactic parse were small compared to the overall performance loss. Then, we conducted a series of experiments on the Brown corpus to get some more information on where the semantic role labeling systems tend to suffer when we go from one genre of text to another, and those results can be summarized as follows:

- There is a significant drop in performance when training and testing on different corpora – for both Treebank and Charniak parses
- In this process the classification task is more disrupted than the identification task.
- There is a performance drop in classification even when training and testing on Brown (compared to training and testing on WSJ)
- The syntactic parser error is not a larger part of the degradation for the case of automatically generated parses.

7.4 General Discussion

The following examples give some insight into the nature of over-fitting to the WSJ corpus. The following output is produced by ASSERT:

(1) SRC enterprise prevented John from [predicate taking] [ARG1 the assignment]

here, “John” is not marked as the agent of “taking”

(2) SRC enterprise prevented [ARG0 John] from [predicate selling] [ARG1 the assignment]

Replacing the predicate “taking” with “selling” corrects the semantic labels, even though the syntactic parse for both sentences is exactly the same. Even using several other predicate in place of “taking” such as “distributing,” “submitting,” etc. give a correct parse. So there is some idiosyncrasy with the predicate “take.”

Further, consider the following set of examples labeled using ASSERT:

(1) [ARG1 The stock] [predicate jumped] [ARG3 from \$ 140 billion to \$ 250 billion] [ARGM-TMP in a few hours of time]

(2) [ARG1 The stock] [predicate jumped] [ARG4 to \$ 140 billion from \$ 250 billion in a few hours of time]

(3) [ARG1 The stock] [predicate jumped] [ARG4 to \$ 140 billion] [ARG3 from \$ 250 billion]

(4) [ARG1 The stock] [predicate jumped] [ARG4 to \$ 140 billion] [ARG3 from \$ 250 billion] [ARGM-TMP after the company promised to give the customers more yields]

(5) [ARG1 The stock] [predicate jumped] [ARG4 to \$ 140 billion] [ARG3 from \$ 250

billion] [ARGM-TMP yesterday]

(6) [ARG1 The stock] [predicate increased] [ARG4 to \$ 140 billion] [ARG3 from \$ 250 billion] [ARGM-TMP yesterday]

(7) [ARG1 The stock] [predicate dropped] [ARG4 to \$ 140 billion] [ARG3 from \$ 250 billion] [ARGM-TMP in a few hours of time]

(8) [ARG1 The stock] [predicate dropped] [ARG4 to \$ 140 billion] [ARG3 from \$ 250 billion within a few hours]

WSJ articles almost always report jump in stock prices by the phrase “to ..” followed by “from ...” and somehow the syntactic parser statistics are tuned to that, and therefore when it faces a sentence like the first one above, two sibling noun phrases are collapsed into one phrase, and so there is only one node in the tree for the two different arguments ARG3 and ARG4 and therefore the role labeler tags it as the more probable of the two and that being ARG3. In the second case, the two noun phrases are identified correctly. The difference in the two is just the transposition of the two words “to” and “from” In the second case, however, the prepositional phrase “in a few hours of time” get attached to the wrong node in the tree, and therefore deleting the node that would have identified the exact boundary of the second argument. Upon deleting the part of the text that is the wrongly attached prepositional phrase, we get the correct semantic role tags in case 3. Now, let's replace this prepositional phrase with a string that happens to be present in the WSJ training data, and see what happens. As seen in example 4, the parser identifies and attaches this phrase correctly and we get a completely correct set of tags. This further strengthens our claim. Even replacing the temporal with a simple one such as “yesterday” maintains the correctness of the tags and also replacing “jumped” with “increased” maintains its correctness. Now, let's

see what happens when the predicate “jump” in example 2 is changed to yet another synonymous predicate – “dropped”. Doing this gives us a correct tagset even though the same syntactic structure is shared between the two, and the prepositional phrase was not attached properly earlier. This shows that just the change of a verb to another changes the syntactic parse to align with the right semantic interpretation. Changing the temporal argument to something slightly different once again causes the parse to fail as seen in 8.

The above examples show that some of the features used in the semantic role labeling, including the strong dependency on syntactic information and therefore the features that are used by the syntactic parser, are too specific to the WSJ. Some obvious possibilities are:

Lexical cues - word usage specific to WSJ.

Verb sub-categorizations - They can vary considerably from one sample of text to another as seen in the examples above and as evaluated in an empirical study by Roland and Jurafsky (1998)

Word senses - domination by unusual word senses (stocks fell)

Topics and entities

While the obvious cause of this behavior is over-fitting to the training data, the question is what to do about it. Two possibilities are:

- Less homogeneous corpora - Rather than using many examples drawn from one source, fewer examples could be drawn from many sources. This would reduce the likelihood of learning idiosyncratic senses and argument structures for predicates.
- Less specific entities - Entity values could be replaced by their class tag (person, organization, location, etc). This would reduce the likelihood of learning

idiosyncratic associations between specific entities and predicates. The system could be forced to use this and more general features.

Both of these manipulations would most likely reduce performance on the training set, and on test sets of the same genre as the training data. But they would likely generalize better. Training on very homogeneous training sets and testing on similar test sets gives a misleading impression of the performance of a system. Very specific features are likely to be given preference in this situation, preventing generalization.

7.5 Nominal Predicates

The argument structure for nominal predicates, when understood in the sense of the nearness of the arguments to the predicate, or through the values of the path that the arguments instantiate, is not usually as complex as the ones for verb predicates. This suggests that the semantics of the words are critical.

This can be better illustrated with an example:

- (1) Napoleon's destruction of the city
- (2) The city's destruction

In the first case, "Napoleon" is the Agent of the nominal argument destruction, but in the second case, the constituent with the same syntactic structure - "the city" is in fact the Theme.

7.6 Considerations for Corpora

Currently, the two primary corpora for semantic role labeling research are PropBank and FrameNet. These two corpora were developed according to very different philosophies. PropBank uses very general arguments whose meanings are generally consistent across predicates, where FrameNet uses role labels specific to a frame (which represents a group of target predicates). FrameNet produces a more specific and precise

representation, where PropBank has better coverage.

The corpora also differ in deciding what instances to annotate. PropBank tags occurrences of verb predicates in an entire corpus, while FrameNet, attempts to find a threshold number of occurrences for each frame. There are advantages and disadvantages to both strategies. The advantage of the former is that all the predicates in a sentence get tagged, making the training data more coherent. This is accompanied by the disadvantage that if a particular predicate, for example, "say" occurs in 70% of the sentences, and that it has only one sense, then the number of examples for that predicate would be disproportionately larger than many other predicates. The advantage of the latter strategy is that the per predicate data tagged can be controlled so as to have near optimal number of examples for training each. In this case, since only part of the corpus is tagged, machine learning algorithms cannot base their decisions by jointly estimating the arguments of all the predicates in a sentence.

We attempted to combine the two corpora to provide more, and more diverse, training data. This proved to be difficult because the segmentation strategies used by the two are different. Efforts are currently underway to provide a mapping between the corpora. PropBank is nearing completion of its attempt at providing frame files, where the core arguments are also tagged with a specific thematic role.

Bibliography

- John Aberdeen, John Burger, David Day, Lynette Hirschman, Patricia Robinson, and Marc Vilain. Mitre: Description of the alembic system as used for muc6. In Proceedings of the Sixth Message Understanding Conference (MUC-6), San Francisco, 1995. Morgan Kaufmann.
- Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. In Proceedings of the 17th International Conference on Machine Learning, pages 9–16. Morgan Kaufmann, San Francisco, CA, 2000.
- Hiyan Alshawi, editor. The Core Language Engine. MIT Press, Cambridge, MA, 1992.
- Michiel Bacchiani, Michael Riley, Brian Roark, and Richard Sproat. MAP adaptation of stochastic grammars. Computer Speech and Language, 20(1):41–68, 2006.
- Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley FrameNet project. In Proceedings of the International Conference on Computational Linguistics (COLING/ACL-98), pages 86–90, Montreal, 1998. ACL.
- Chris Barker and David Dowty. Non-verbal thematic proto-roles. In Proceedings of North-Eastern Linguistics Conference (NELS-23), Amy Schafer, ed., GSLA, Amherst, pages 49–62, 1992.
- R. E. Barlow, D. J. Bartholomew, J. M. Bremner, and H. D. Brunk. Statistical Inference under Order Restrictions. Wiley, New York, 1972.
- Daniel M. Bikel, Richard Schwartz, and Ralph M. Weischedel. An algorithm that learns what’s in a name. Machine Learning, 34:211–231, 1999.
- Don Blaheta and Eugene Charniak. Assigning function tags to parsed text. In Proceedings of the 1st Annual Meeting of the North American Chapter of the ACL (NAACL), pages 234–240, Seattle, Washington, 2000.
- Daniel C. Bobrow. Natural language input for a computer problem solving system. In Marvin Minsky, editor, Semantic Information Processing, pages 146–226. MIT Press, Cambridge, MA, 1968.
- Daniel G. Bobrow. Natural language input for a computer problem solving system. Technical report, Cambridge, MA, USA, 1964.

- Xavier Carreras and Lluís Màrquez. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005), pages 152–164, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/W/W05/W05-0620>.
- Eugene Charniak. A maximum-entropy-inspired parser. In Proceedings of the 1st Annual Meeting of the North American Chapter of the ACL (NAACL), pages 132–139, Seattle, Washington, 2000.
- Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and maxent discriminative reranking. In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05), pages 173–180, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/P/P05/P05-1022>.
- John Chen and Owen Rambow. Use of deep linguistics features for the recognition and labeling of semantic arguments. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Sapporo, Japan, 2003.
- Michael Collins. Three generative, lexicalised models for statistical parsing. In Proceedings of the 35th Annual Meeting of the ACL, pages 16–23, Madrid, Spain, 1997.
- Michael John Collins. Head-driven Statistical Models for Natural Language Parsing. PhD thesis, University of Pennsylvania, Philadelphia, 1999.
- K. Daniel, Y. Schabes, M. Zaidel, and D. Egedi. A freely available wide coverage morphological analyzer for english. In Proceedings of the 14th International Conference on Computational Linguistics (COLING-92), Nantes, France., 1992.
- David R. Dowty. Thematic proto-roles and argument selction. Language, 67(3):547–619, 1991.
- Charles J. Fillmore and Collin F. Baker. FrameNet: Frame semantics meets the corpus. In Poster presentation, 74th Annual Meeting of the Linguistic Society of America, January 2000.
- Michael Fleischman, Namhee Kwon, and Eduard Hovy. Maximum entropy models for framenet classification. In Proceedings of the Empirical Methods in Natural Language Processing, , Sapporo, Japan, 2003.
- Dean P. Foster and Robert A. Stine. Variable selection in data mining: building a predictive model for bankruptcy. Journal of American Statistical Association, 99: 303–313, 2004.
- Dan Gildea and Julia Hockenmaier. Identifying semantic roles using combinatory categorial grammar. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Sapporo, Japan, 2003.
- Daniel Gildea. Corpus variation and parser performance. In In Proceedings of Empirical Methors in Natural Language Processing (EMNLP), 2001.

- Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. Computational Linguistics, 28(3):245–288, 2002.
- Daniel Gildea and Martha Palmer. The necessity of syntactic parsing for predicate argument recognition. In Proceedings of the 40th Annual Conference of the Association for Computational Linguistics (ACL-02), Philadelphia, PA, 2002.
- Bert F. Green, K. Wolf, Alice, Chomsky, Carol, Laughery, and Kenneth. Baseball: an automatic question answerer. In Proceedings of the Western Joint Computer Conference, pages 219–224, May 1961.
- Bert F. Green, K. Wolf, Alice, Chomsky, Carol, Laughery, and Kenneth. Baseball: an automatic question answerer. In Margaret King, editor, Computers and Thought. MIT Press, Cambridge, MA, 1963.
- Ralph Grishman, Catherine Macleod, and John Sterling. New york university: Description of the proteus system as used for muc-4. In Proceedings of the Fourth Message Understanding Conference (MUC-4), 1992.
- Kadri Hacioglu. A lightweight semantic chunking model based on tagging. In Proceedings of the Human Language Technology Conference /North American chapter of the Association of Computational Linguistics (HLT/NAACL), Boston, MA, 2004a.
- Kadri Hacioglu. Semantic role labeling using dependency trees. In Proceedings of COLING-2004, Geneva, Switzerland, 2004b.
- Kadri Hacioglu and Wayne Ward. Target word detection and semantic role chunking using support vector machines. In Proceedings of the Human Language Technology Conference, Edmonton, Canada, 2003.
- Kadri Hacioglu, Sameer Pradhan, Wayne Ward, James Martin, and Dan Jurafsky. Shallow semantic parsing using support vector machines. Technical Report TR-CSLR-2003-1, Center for Spoken Language Research, Boulder, Colorado, 2003.
- Kadri Hacioglu, Sameer Pradhan, Wayne Ward, James Martin, and Daniel Jurafsky. Semantic role labeling by tagging syntactic chunks. In Proceedings of the 8th Conference on CoNLL-2004, Shared Task – Semantic Role Labeling, 2004.
- George E. Heidorn. English as a very high level language for simulation programming. In Proceedings of Symposium on Very High Level Languages, Sigplan Notices, pages 91–100, 1974.
- C. Hewitt. PLANNER: A language for manipulating models and proving theorems in a robot. Technical report, Cambridge, MA, USA, 1970.
- Graeme Hirst. A foundation for semantic interpretation. In Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics, pages 64–73, Cambridge, MA, 1983.

- Jerry R. Hobbs, Douglas Appelt, John Bear, David Israel, Megumi Kameyama, Mark E. Stickel, and Mabry Tyson. FASTUS: A cascaded finite-state transducer for extracting information from natural-language text. In Emmanuel Roche and Yves Schabes, editors, Finite-State Language Processing, pages 383–406. MIT Press, Cambridge, MA, 1997.
- Thomas Hofmann and Jan Puzicha. Statistical models for co-occurrence data. Memo, Massachusetts Institute of Technology Artificial Intelligence Laboratory, Feb 1998.
- Richard D. Hull and Fernando Gomez. Semantic interpretation of nominalizations. In Proceedings of the Thirteenth National Conference on Artificial Intelligence, Portland, Oregon, pages 1062–1068, 1996.
- Ray Jackendoff. Semantic Interpretation in Generative Grammar. MIT Press, Cambridge, Massachusetts, 1972.
- Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In Proceedings of the European Conference on Machine Learning (ECML), 1998.
- Ulrich H G Kressel. Pairwise classification and support vector machines. In Bernhard Scholkopf, Chris Burges, and Alex J. Smola, editors, Advances in Kernel Methods. The MIT Press, 1999.
- Henry Kučera and W. Nelson Francis. Computational analysis of present-day American English. Brown University Press, Providence, RI, 1967.
- Taku Kudo and Yuji Matsumoto. Use of support vector learning for chunk identification. In Proceedings of the 4th Conference on CoNLL-2000 and LLL-2000, pages 142–144, 2000.
- Taku Kudo and Yuji Matsumoto. Chunking with support vector machines. In Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2001), 2001.
- Maria Lapata. The disambiguation of nominalizations. Computational Linguistics, 28 (3):357–388, 2002.
- LDC. The AQUAINT Corpus of English News Text, Catalog no. LDC2002t31, 2002. URL <http://www.ldc.upenn.edu/Catalog/docs/LDC2002T31/>.
- Dekang Lin. Automatic retrieval and clustering of similar words. In Proceedings of the International Conference on Computational Linguistics (COLING/ACL-98), Montreal, Canada, 1998a.
- Dekang Lin. Dependency-based evaluation of MINIPAR. In In Workshop on the Evaluation of Parsing Systems, Granada, Spain, 1998b.
- Dekang Lin and Patrick Pantel. Discovery of inference rules for question answering. Natural Language Engineering, 7(4):343–360, 2001.

- Robert K. Lindsay. Inferential memory as the basis of machines which understand natural language. In Margaret King, editor, Computers and Thought. MIT Press, Cambridge, MA, 1963.
- Rey-Long Liu and Von-Wun Soo. An empirical study on thematic knowledge acquisition based on syntactic clues and heuristics. In Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics, pages 243–250, Ohio State University, Columbus, Ohio, 1993.
- Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. Journal of Machine Learning Research, 2(Feb): 419–444, 2002.
- Catherine Macleod, Ralph Grishman, Adam Meyers, Leslie Barrett, and Ruth Reeves. Nomlex: A lexicon of nominalizations, 1998.
- David Magerman. Natural Language Parsing as Statistical Pattern Recognition. PhD thesis, Stanford University, CA, 1994.
- Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The Penn Treebank: Annotating predicate argument structure, 1994a.
- Mitchell P. Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The Penn Treebank: Annotating predicate argument structure. In ARPA Human Language Technology Workshop, pages 114–119, Plainsboro, NJ, 1994b. Morgan Kaufmann.
- James L. McClelland and Alan H. Kawamoto. Parallel distributed processing. In J. L. McClelland and D. E. Rumelhart, editors, Mechanisms of Sentence Processing: Assigning roles to constituents of sentences. MIT Press, 1986.
- David McClosky, Eugene Charniak, and Mark Johnson. Reranking and self-training for parser adaptation. In Proceedings of the Annual Meeting of the Association for Computational Linguistics (COLING-ACL'06), Sydney, Australia, July 2006a. Association for Computational Linguistics.
- David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In Proceedings of the Human Language Technology Conference of the NAACL, Main Conference, pages 152–159, New York City, USA, June 2006b. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/N/N06/N06-1020>.
- Martha Palmer, Carl Weir, Rebecca Passonneau, and Tim Finin. The kernel text understanding system. Artificial Intelligence, 63:17–68, October 1993. Special Issue on Text Understanding.
- Martha Palmer, Dan Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. Computational Linguistics, pages 71–106, 2005a.
- Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. Computational Linguistics, 31(1):71–106, 2005b.

- John Platt. Probabilities for support vector machines. In A. Smola, P. Bartlett, B. Scholkopf, and D. Schuurmans, editors, Advances in Large Margin Classifiers. MIT press, Cambridge, MA, 2000.
- Sameer Pradhan, Valerie Krugler, Wayne Ward, James Martin, and Dan Jurafsky. Using semantic representations in question answering. In Proceedings of the International Conference on Natural Language Processing (ICON-2002), pages 195–203, Bombay, India, 2002.
- Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James Martin, and Dan Jurafsky. Support vector learning for semantic argument classification. Technical Report TR-CSLR-2003-3, Center for Spoken Language Research, Boulder, Colorado, 2003a.
- Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James Martin, and Dan Jurafsky. Semantic role parsing: Adding semantic structure to unstructured text. In Proceedings of the International Conference on Data Mining (ICDM 2003), Melbourne, Florida, 2003b.
- Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Dan Jurafsky. Shallow semantic parsing using support vector machines. In Proceedings of the Human Language Technology Conference/North American chapter of the Association of Computational Linguistics (HLT/NAACL), Boston, MA, 2004.
- Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James Martin, and Dan Jurafsky. Support vector learning for semantic argument classification. Machine Learning Journal, 60(1):11–39, 2005a.
- Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Dan Jurafsky. Semantic role labeling using different syntactic views. In Proceedings of the Association for Computational Linguistics 43rd annual meeting (ACL-2005), Ann Arbor, MI, 2005b.
- J. Ross Quinlan. Induction of decision trees. Machine Learning, 1(1):81–106, 1986.
- Ross Quinlan. Data Mining Tools See5 and C5.0, 2003. <http://www.rulequest.com>.
- L. A. Ramshaw and M. P. Marcus. Text chunking using transformation-based learning. In Proceedings of the Third Annual Workshop on Very Large Corpora, pages 82–94. ACL, 1995.
- Adwait Ratnaparkhi. A maximum entropy part-of-speech tagger. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 133–142, University of Pennsylvania, May 1996. ACL.
- Ellen Riloff. Automatically constructing a dictionary for information extraction tasks. In Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI), pages 811–816, Washington, D.C., 1993.
- Ellen Riloff. Automatically generating extraction patterns from untagged text. In Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI), pages 1044–1049, 1996.

- Ellen Riloff and Rosie Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI), pages 474–479, 1999.
- Douglas Roland and Daniel Jurafsky. How verb subcategorization frequencies are affected by corpus choice. In Proceedings of COLING/ACL, pages 1122–1128, Montreal, Canada, 1998.
- Joao Luis Garcia Rosa and Edson Francozo. Hybrid thematic role processor: Symbolic linguistic relations revised by connectionist learning. In IJCAI, pages 852–861, 1999. URL citeseer.nj.nec.com/rosa99hybrid.html.
- Wolfgang Samlowski. Case grammar. In Eugene Charniak and Yorick Wilks, editors, Computational Semantics: An Introduction to Artificial Intelligence and Natural Language Comprehension. North Holland Publishing Company, 1976.
- Roger C. Schank. Conceptual dependency: a theory of natural language understanding. Cognitive Psychology, 3:552–631, 1972.
- Roger C. Schank, Neil M. Goldman, Charles J. Rieger, and Chistopher Riesbeck. MARGIE: Memory Analysis Response Generation, and Inference on English. In Proceedings of the International Joint Conference on Artificial Intelligence, pages 255–261, 1973.
- Fabrizio Sebastiani. Machine learning in automated text categorization. ACM Computing Surveys, 34(1):1–47, 2002.
- Robert F. Simmons. Answering english questions by computer: a survey. Commun. ACM, 8(1):53–70, 1965. ISSN 0001-0782.
- Norman Sondheimer, Ralph Weischedel, and Robert Bobrow. Semantic interpretation using kl-one. In Proceedings of the 10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics, pages 101–107, 1984.
- Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. Using predicate-argument structures for information extraction. In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, Sapporo, Japan, 2003.
- G. J. Sussman, T. Winograd, and E. Charniak. microPLANNER reference manual. Technical report, Cambridge, MA, USA, 1971.
- David L. Waltz. The state of the art in natural-language understanding. In Wendy G. Lehnert and Martin H. Ringle, editors, Strategies for Natural Language Processing, pages 3–32. Lawrence Erlbaum, New Jersey, 1982.
- David Scott Warren and Joyce Friedman. Using semantics in non-context-free parsing of montague grammar. Computational Linguistics, 8(3-4):123–138, 1982.
- Joseph Weizenbaum. ELIZA – A computer program for the study of natural language communication between man and machine. Communications of the ACM, 9(1):36–45, January 1966.

- J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for svms. Advances in Neural Information Processing Systems (NIPS), 13: 668–674, 2001.
- Terry Winograd. Understanding Natural Language. Academic Press, New York, 1972.
- Terry Winograd. Procedures as a representation for data in a computer program for understanding natural language. Technical Report AI Technical Report 235, MIT, 1971.
- William Woods. Semantics for Question Answering System. PhD thesis, Harvard University, 1967.
- William Woods. Progress in natural language understanding: an application to lunar geology. In Proceedings of AFIPS, volume 42, pages 441–450, 1973.
- William A. Woods. Transition network grammars for natural language analysis. Communications of the ACM, 13(10):591–606, 1970.
- William A. Woods. Semantics and quantification in natural language question answering. In M. Yovits, editor, Advances in Computers, pages 2–64. Academic, New York, 1978.
- William A. Woods. Lunar rocks in natural English: Explorations in natural language question answering. In Antonio Zampolli, editor, Linguistic Structures Processing, pages 521–569. North Holland, Amsterdam, 1977.
- Nianwen Xue and Martha Palmer. Calibrating features for semantic role labeling. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Barcelona, Spain, 2004.
- Roman Yangarber and Ralph Grishman. Nyu: Description of the proteus/pet system as used for muc-7 st. In Proceedings of the Sixth Message Understanding Conference (MUC-7), Varginia, 1998.

Appendix A

Temporal Words

year	july	october	typically	generally
yesterday	weeks	minutes	hour	jan
years	previously	eventually	ended	future
quarter	end	immediately	november	term
week	june	night	shortly	early;year
months	early	finally	earlier;month	
time	past	dec	decade	
month	period	thursday	tomorrow	
friday	april	recent	frequently	
ago	nov	aug	weekend	
recently	long	morning	hours	
oct	march	initially	temporarily	
today	late	longer	fall	
sept	years;ago	afternoon	annually	
september	tuesday	past;years	february	
day	wednesday	fourth;quarter	mid	
earlier	summer	spring	half	
august	earlier;year	year;ago	recent;years	
monday	january	moment	fourth	
days	december	year;earlier	year;end	
