Support Vector Learning for Semantic Argument Classification[*]

Sameer Pradhan, Kadri Hacioglu, Valerie Krugler,
Wayne Ward, James H. Martin, Daniel Jurafsky

TR-CSLR-2003-03

Center for Spoken Language Research,
University of Colorado, Boulder, CO 80303
`sameer.pradhan@colorado.edu`

Abstract

The natural language processing community has recently experienced a growth of interest in domain independent shallow semantic parsing – the process of assigning a WHO did WHAT to WHOM, WHEN, WHERE, WHY, HOW etc. structure to plain text. This process entails identifying groups of words in a sentence that represent these semantic arguments and assigning specific labels to them. It could play a key role in NLP tasks like Information Extraction, Question Answering and Summarization. We propose a new machine learning algorithm for semantic role parsing, extending the work of Gildea and Jurafsky (2002), Surdeanu et al. (2003) and others. Our algorithm is based on Support Vector Machines which we show give large improvement in performance over earlier classifiers. We show performance improvements through a number of new features designed to improve generalization to unseen data, such as automatic clustering of verbs. We also report on various analytic studies examining which features are most important, comparing our classifier to other machine learning algorithms in the literature, and testing its generalization to new test set from different genre. On the task of assigning semantic labels to the PropBank (Kingsbury et al., 2002) corpus, our final system has a precision of 82% and a recall of 73%, which are the best results currently reported for this task. Finally, we explore a completely different architecture which does not requires a deep syntactic parse. We reformulate the task as a combined chunking and classification problem, thus allowing our algorithm to be applied to new languages or genres of text for which statistical syntactic parsers may not be available.

keywords: *shallow semantic parsing support vector machines*

[*]This research was partially supported by the ARDA AQUAINT program via contract OCG4423B and by the NSF via grant IIS-9978025

1 Introduction

Automatic, accurate and wide-coverage techniques that can annotate naturally occurring text with semantic argument structure can facilitate the discovery of patterns of information in large text collections Wallis and Nelson (2001); Hearst (1999). Shallow semantic parsing – the process of assigning a simple WHO did WHAT to WHOM, WHEN, WHERE, WHY, HOW, etc. structure to sentences in text, is the process of producing such a markup. More specifically, when presented with a sentence, a parser should, for each predicate in the sentence, identify and label the predicate's semantic arguments. This process entails identifying groups of words in a sentence that represent these semantic arguments and assigning specific labels to them. This notion of shallow semantic parsing, or case role analysis, has a long history in the computational linguistics literature. For a short review of this history, the reader is referred to Jurafsky and Martin (2000)

In recent work, a number of researchers have cast this problem as a tagging, problem and have applied various supervised machine learning techniques to it Gildea and Jurafsky (2000); Blaheta and Charniak (2000); Gildea and Jurafsky (2002); Gildea and Palmer (2002); Gildea and Hockenmaier (2003); Surdeanu et al. (2003); Chen and Rambow (2003); Fleischman and Hovy (2003); Hacioglu and Ward (2003); Thompson et al. (2003); Pradhan et al. (2003). In this paper, we report on a series of experiments exploring this approach.

For the initial experiments, we adopted the approach described by Gildea and Jurafsky (2002) and evaluated a series of modifications to improve its performance. In the experiments reported here we first replaced their statistical classification algorithm with one that uses Support Vector Machines and then added some more features to the existing feature set. All of the experiments we report were conducted on PropBank, an approximately 300k-word corpus annotated with semantic arguments of verbs. We evaluated the system on three main tasks: identifying words/phrases that represent semantic arguments, independently classifying words/phrases known to be semantic arguments into the specific categories, and the combined task of identifying the arguments and then assigning respective labels to them. We evaluated results using both gold-standard syntactic parses, and actual parses from the Charniak parser.

A fundamental assumption in this architecture is the presence of a full syntactic parse, which provides the bulk of the features used by the machine learning algorithm. This limits its applicability to the language and genre of text for which statistical syntactic parsers are available, or can be trained. Therefore, we re-formulated the task as a combined chunking and classification problem retaining the same classifier and replacing some features that depend on the hierarchical structure of the syntactic parse, with ones derived from a flat chunked representation.

Finally, to test the sensitivity of both sets of features to change in corpus, we manually tagged about 400 sentences from the AQUAINT corpus (LDC, 2002), which is a collection of text from the New York Times Inc., Associated Press Inc., and Xinhua News Service, compared to the Wall Street Journal corpus, on which the system is trained. Perhaps not surprisingly, there is a significant drop in performance on this new source of text.

Section 2 introduces the semantic annotation schema and corpora used for the task. In Section 3 we will discuss the semantic parsing system that makes use of features extracted from a full syntactic parse, and which assumes that the semantic argument boundaries align with those of the syntactic parser. In Section 4 we will discuss in detail the system that does not depend on a full syntactic parse, by formulating the semantic parsing problem as a chunking and classification task.

2 Semantic Annotation and Corpora

Two corpora are available for developing and testing semantic argument annotation – FrameNet[1] (Baker et al., 1998) and PropBank[2] (Kingsbury et al., 2002). FrameNet uses predicate specific labels such as Judge and Judgee. PropBank uses predicate independent labels – Arg0, Arg1, etc. In this paper, we will be reporting on results using PropBank, a 300k-word corpus in which predicate argument relations are marked for almost all occurrences of non-copula verbs in the Wall Street Journal (WSJ) part of the Penn TreeBank (Marcus et al., 1994). The arguments of a verb are labeled sequentially from Arg0 to Arg5, where Arg0 is the Proto-Agent (usually the subject of a transitive verb) Arg1 is the Proto-Patient (usually its direct object), etc. PropBank does attempt to treat semantically related verbs consistently. In addition to these "core arguments," additional "adjunctive arguments," referred to as ArgMs are also marked. Some examples are ArgM-Loc, for locatives, and ArgM-Tmp, for temporals. We will refer to these as ArgMs. Table 1 shows the argument labels associated with the predicate *operate* in PropBank.

Table 1: Argument labels associated with the predicate *operate* (sense: work) in the PropBank corpus.

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

Following is an example structure extracted from PropBank corpus. The syntax tree representation along with the argument labels is shown in Figure 1.

[1] http://www.icsi.berkeley.edu/~framenet/
[2] http://www.cis.upenn.edu/~ace/

[**ARG0** It] [**predicate** operates] [**ARG1** stores] [**ARGM−LOC** mostly in Iowa and Nebraska].

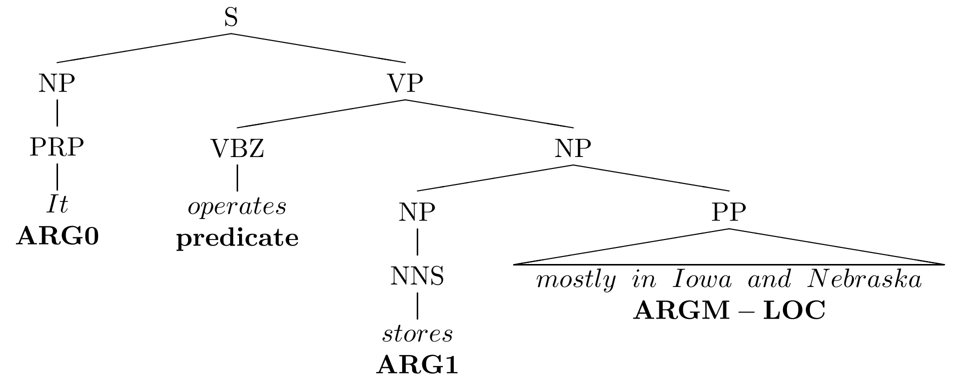


Figure 1: Syntax tree for a sentence illustrating the PropBank tags

Table 2: List of adjunctive arguments in PropBank – ArgMs

Tag	Description	Examples
ArgM-Loc	Locative	*the museum, in Westborough, Mass.*
ArgM-Tmp	Temporal	*now, by next summer*
ArgM-Mnr	Manner	*heavily, clearly, at a rapid rate*
ArgM-Dir	Direction	*to market, to Bangkok*
ArgM-Cau	Cause	*In response to the ruling*
ArgM-Dis	Discourse	*for example, in part, Similarly*
ArgM-Ext	Extent	*at $38.375, 50 points*
ArgM-Prp	Purpose	*to pay for the plant*
ArgM-Neg	Negation	*not, n't*
ArgM-Mod	Modal	*can, might, should, will*
ArgM-Rec	Reciprocals	*each other*
ArgM-Prd	Secondary Predication	*to become a teacher*
ArgM	Bare ArgM	*with a police escort*
ArgM-Adv	Adverbials	*(none of the above)*

The adjunctive arguments are shown in Table 2. Thus for example, Arg0 usually corresponds to a semantic agent, while Arg1 typically corresponds to the argument most affected by the action, usually referred to as the theme.

All experiments in this paper are performed on the July 2002 release of PropBank. PropBank was constructed by assigning semantic arguments to constituents of the "gold-standard" TreeBank parses. The data comprise several sections of the Wall Street Journal, and we follow the standard convention of using Section-23 data as the test set. Section-02 to Section-21 were used for training. The training set comprises about 51,000 sentences, instantiating about 132,000 arguments, and the test set comprises 2,700 sentences instantiating about 7,000 arguments.

3 Constituent-by-Constituent Semantic Parsing

3.1 Problem Description

The problem of shallow semantic parsing can be viewed as a three step process. First, is the process of identifying the predicate whose arguments we plan to identify. In our case, this is any non-copula verb in the sentence. The second step is to identify words or phrases that represent the semantic arguments of that predicate. The third step is to assign specific argument class labels to those words or phrases. This process can be evaluated on three distinct tasks:

- *Argument Identification* – This is the process of identifying parse constituents in the sentence that represent valid semantic arguments of a given predicate.
- *Argument Classification* – Given constituents known to represent valid arguments of a predicate, assign the appropriate argument labels to them.
- *Argument Identification and Classification* – A combination of the above two tasks, where the system first identifies probable constituents that represent arguments of a predicate, and then assigns them the most likely argument labels.

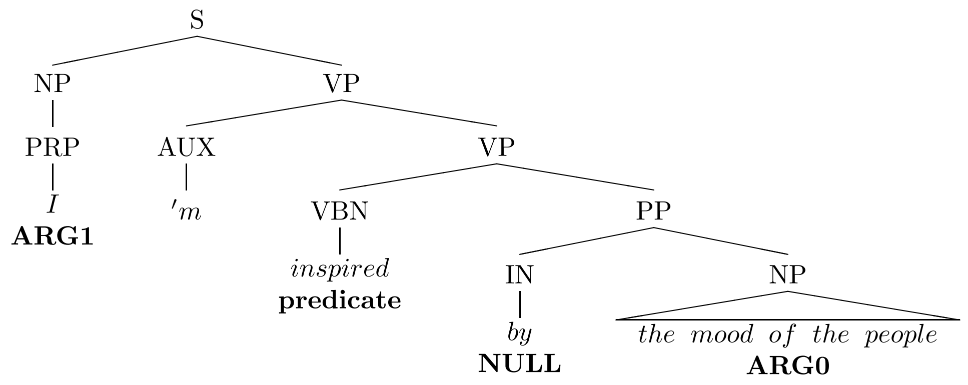


Figure 2: A sample sentence from the PropBank corpus

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, 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.

3.2 Baseline Features

Our baseline system uses the same set of features introduced by Gildea and Jurafsky (2002): predicate, path, phrase type, position, voice, head word, and verb sub-categorization. All these features (except one – the predicate) are extracted from the syntactic parse tree of the sentence. Some of the features, viz., predicate, voice and verb sub-categorization are shared by all the nodes in the tree. All the others change with the constituent under consideration.

- **Predicate** – The predicate itself is used as a feature after performing morphological stemming using the XTAG morphology database[3] (Daniel et al., 1992).

- **Path** – The syntactic path through the parse tree from the parse constituent to the predicate being classified. For example, in Figure 3, the path from ARG0 – "The lawyers" to the predicate "went", is represented with the string NP$\uparrow$S$\downarrow$VP$\downarrow$VBD. $\uparrow$ and $\downarrow$ represent upward and downward movement in the tree respectively.

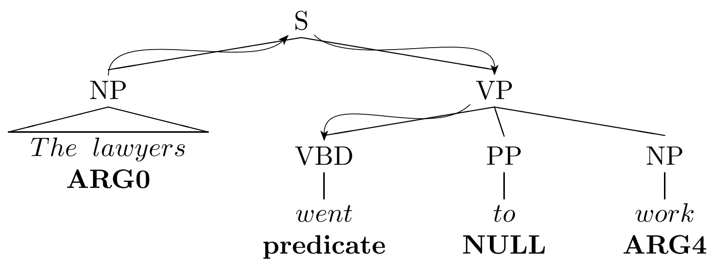


Figure 3: Illustration of path NP$\uparrow$S$\downarrow$VP$\downarrow$VBD

- **Phrase Type** – This is the syntactic category (NP, PP, S, etc.) of the phrase corresponding to the semantic argument.

- **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. We run a set of hand-written `tgrep` expressions on the syntax

[3]`ftp://ftp.cis.upenn.edu/pub/xtag/morph-1.5/morph-1.5.tar.gz`

tree to identify passive voiced predicates. In approximately 11% of the sentences in PropBank, the predicate had a passive instantiation.

- **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). We perform a case-folding operation 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 3, the sub-categorization for the predicate "went" is VP→VBD-PP-NP.

3.3 Classifier

Support Vector Machines (SVMs) have been shown to perform well on text classification tasks, where data is represented in a high dimensional space using sparse feature vectors (Joachims, 1998; Kudo and Matsumoto, 2000; Lodhi et al., 2002). As we have defined it, semantic parsing is a multi-class classification problem. However, SVMs are binary classifiers. It is well known that a multi-class problem can be reduced to a number of binary-class problems. There are two common approaches for extending SVMs to multi-class classification problems (Allwein et al., 2000). The first is known as the Pairwise approach, where a separate binary classifier is trained for each of the class pairs and their outputs are combined to predict the classes. This approach requires the training of $\frac{N(N-1)}{2}$ binary classifiers. The second, known as the One *vs* All (OVA) approach, involves training n classifiers for a n-class problem. The classifiers are trained to discriminate between examples of each class, and those belonging to all other classes combined. During testing, the classifier scores on an example are combined to predict its class label. Among these two approaches, there is a trade-off between the number of classifiers to be trained and the data used in training each classifier.

While some experiments have been reported that the pairwise approach outperforms the OVA approach (Krebel, 1999), our initial experiments show better performance for OVA. Therefore, in this paper, all results are using the OVA classification approach.

3.4 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 80% 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 > 0.95) are pruned from the training set. This is accompanied by a very negligible ($< 0.5\%$) pruning of nodes that are NON-NULL.

2. The remaining training data is used to train a OVA classifier that contains the all the classes along with the 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, we do not perform any filtering of NULL nodes in a first pass. Instead, 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 classifier in a first pass, as we do in the training process. Pseudocode for the training and testing algorithms is shown in Figures 4 and 5 respectively.

For our experiments, we used tinySVM[4] along with YamCha[5] (Kudo and Matsumoto, 2000, 2001) as the SVM training and test software. The system uses a polynomial kernel with degree 2; the cost per unit violation of the margin, C=1; and, tolerance of the termination criterion, e=0.001.

[4] `http://cl.aist-nara.ac.jp/~talus-Au/software/TinySVM/`
[5] `http://cl.aist-nara.ac.jp/~taku-Au/software/yamcha/`

```
procedure $TrainSemanticParser(TrainingData)$
for $sentence \in TrainingData$ do
  step Generate a full syntactic parse of the sentence/Use available
       "gold-standard" parse
  step Extract the required set of features
  step Label the nodes NULL or NON-NULL
  step Add to $TrainingExamples$
end for

step Train a NULL vs NON-NULL SVM classifier using $TrainingExamples$
step Randomly select $HeldOutTrainingExamples$

for $class \in$ {NULL, NON-NULL} do
  step Generate raw SVM scores for $HeldOutTrainingExamples$
  step Fit a sigmoid function to class scores of all examples in
       $HeldOutTrainingExamples$
end for

step $UnprunedTrainingExamples = \{\}$
for $example \in TrainingExamples$ do
  step Use the SVM classifier to generate raw SVM scores
       for each $class \in$ {NULL, NON-NULL}
  step Use the sigmoid fitted to class to convert raw SVM scores to
       probabilities
  if $P(class ==$NULL$) > threshold$ then
    step Prune example
  else
    if (class ==NON-NULL) then
      step Replace the class label with the actual argument class of example
    end if
    step Add example to $UnprunedTrainingExamples$
  end if
end for

for $class \in$ {NULL, ARG0-5, ARGMS} do
  step Train a class vs Other classifier using $UnprunedTrainingExamples$
end for
```

Figure 4: The baseline two-stage SVM training algorithm

```
procedure SemanticParse(Sentence)
step Generate a full syntactic parse of the Sentence
step Identify all the verb predicates
for predicate ∈ Sentence do
  step Extract a set of features for each node in the tree relative
       to the predicate
  step Classify each node as NULL or as one of the PropBank arguments
       using the maximum scoring one out of the 23 ONE vs ALL SVM
       classifiers
  step Return the best non-overlapping sequence of NON-NULL nodes
end for
```

Figure 5: The semantic parsing algorithm

Table 3: Baseline performance on all the three tasks using "gold-standard" parses

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

3.5 Baseline System Performance

Table 3 shows the baseline performance numbers on all the three tasks mentioned earlier in Section 3.1 using the PropBank corpus with "gold-standard" parses, i.e., the parses from the hand-corrected Penn TreeBank[6] The set features listed in Section 3.2 were used. For the argument identification and the combined identification and classification tasks, we report the precision (P), recall (R) and the F_β[7] scores, and for the argument classification task we report the classification accuracy (A). This test set and all test sets, unless noted otherwise are Section-23 of PropBank.

3.6 System Improvements

3.6.1 Disallowing Overlaps

The system as described above might label two constituents even if they overlap in words. This is a problem since we would like to have each word to belong

[6]We report on gold standard parse values because it is easier to compare with results from the literature; we will also report on performance using an actual errorful parser.

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

to at most one semantic argument. We can solve this problem using the fact that overlapping arguments were not allowed in the PropBank training set. We propose to choose among the overlapping constituents by retaining the one for which the SVM has the highest confidence, and labeling the others Null. The probabilities obtained by applying the sigmoid function to the raw SVM scores are used as the measure of confidence. Table 4 shows the performance of the parser on the task of identifying and labeling semantic arguments using the "gold-standard" parses. In this system, the overlap-removal decisions are taken independently of each other. In the future, we plan to perform an optimal search using additional information such as the language model scores of the role sequence, along with the top n hypotheses for a constituent, to identify the most likely sequence of arguments.

Table 4: Improvements on disallowing overlapping constituents

	P (%)	R (%)	$F_{\beta=1}$	
Baseline	83.3	78.5	80.8	
No Overlaps	85.4	78.1	81.6	($\chi^2; p < 0.01$)

3.6.2 New Features

We tested four features, two of which were obtained from the literature – named entities in constituents and head word part of speech. The other two are novel features: predicate verb cluster and partial-path.

Named Entities in Constituents

Surdeanu et al. (2003) reported a performance improvement on classifying the semantic role of the constituent by relying on the presence of a named entity in the constituent. We expect 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 a closed set of locative or temporal cues. We evaluated a similar use of named entities in our system. We tagged 7 named entities (Person, Organization, Location, Percent, Money, Time, Date) using Identifinder (Bikel et al., 1999) and added them as 7 binary features. Each of these features is true if the respective named entity is contained in the constituent.

We found that adding this feature 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 particular 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. The same would not be true for the role labeling task. We then tried using this as a feature solely in the task of classifying constituents known to represent arguments, using features extracted from "gold-standard" parses. On this task, overall classification accuracy increased from 87.1% to 88.1% ($\chi^2; p = 0.08$). As expected, the most significant improvement was for adjunct arguments like temporals (ARGM-TMP) and locatives (ARGM-LOC) as shown in Table 5. Both improvements were found to be significant at the 0.05 probability level using the χ^2-test. 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.

Table 5: Improvement using named entities

	ARGM-LOC			ARGM-TMP		
	P (%)	R (%)	$F_{\beta=1}$	P (%)	R (%)	$F_{\beta=1}$
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

Head Word Part of Speech

Surdeanu et al. (2003) showed that adding of the 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. Table 6 shows performance on the argument identification task with and without this feature. 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. On the task of argument classification, adding this feature increased the accuracy from 87.1% to 88.6% ($\chi^2; p < 0.05$).

Table 6: Improvements on adding head word part of speech

	P (%)	R (%)	$F_{\beta=1}$
Baseline	91.2	89.6	90.4
With Head Word POS	92.2	90.5	91.3 ($\chi^2; p < 0.05$))

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 using predicate cluster as a feature. 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 (1998). The verbs were clustered into 64 classes using the probabilistic co-occurrence model of Hofmann (Hofmann and Puzicha, 1998).

We then use the verb class of the current predicate as a feature. The performance improvement on the task of identifying and labeling semantic arguments using the "gold-standard" PropBank corpus is shown in Table 7. This increase in recall, at very little cost of precision, was obtained despite of the fact that more than 99% of the predicates in the PropBank test set are present in the training set. Presumably, it would be more salient in cases where there are no training statistics available for the predicate.

Table 7: Improvement on adding verb-cluster

	P (%)	R (%)	$F_{\beta=1}$
Baseline	83.3	78.5	80.8
With Cluster	82.6	80.0	81.8 $(\chi^2; p < 0.01)$

Generalizing the Path Feature

As we will see in Section 3.9, 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, we tried generalizing the path 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 3 is NP↑S.

None of these features significantly affected NULL *vs* NON-NULL classification. The "Partial-Path" feature improved the argument classification performance for "gold-standard" parses from 87.1% to 88.3% $(\chi^2; p < 0.05)$.

3.6.3 Alternative Pruning Strategies

In the baseline system architecture, we opted for the two step training strategy shown in Figure 4. To evaluate possible performance degradation accompanying the savings in training time, we tested two other pruning strategies. The three strategies in order of increased training time are:

- **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.
- **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 > 0.95) of being NULL in a first pass and then trains ONE *vs* ALL classifiers on the remaining data *including* the NULL class.
- **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.

Table 8 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 ($\chi^2; p = 0.08$). 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 80% of the nodes in a tree are NULL arguments, and iii) We have to optimize only one classifier on the entire data. We continued to use the *two-pass soft-prune* strategy

Table 8: Comparing pruning strategies

	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

3.7 Best System Performance

Table 9 shows the collective impact of all the modifications on the system, and is our current best performing system.

Table 9: Best system performance on all tasks using gold-standard parses.

Classes	Task	Gold			
		P (%)	R (%)	$F_{\beta=1}$	A (%)
ALL ARGUMENTS	Identification	94.2	89.7	91.9	
	Classification	-	-	-	88.9
	Identification + Classification	86.1	80.9	83.4	

3.8 Using Automatic Parses

Thus far, we have reported results using hand-corrected "gold-standard" parses. In real-word applications, the system will have to extract features from an automatically generated parse. To evaluate this scenario, we used the Charniak parser (Chaniak, 2001) to generate parses for PropBank training and test data.

In these generated parses, there are about 6% arguments whose boundaries do not align exactly with any of the hand-generated phrase boundaries. We tried to recover training information in such cases by automatically correcting the slight misalignment caused by a class of parser errors – as done before by Gildea and Jurafsky (2002). In order to align the automatically generated constituents with the hand generated ones, words were dropped one by one from the right end of a hand-generated constituent until it matched one of the automatically generated constituents, with which it was then aligned. This procedure was followed for both training and test data. Table 10 shows the performance degradation when automatically generated parses are used. Interestingly, the classification accuracy is the least affected by the transition from "gold-standard" parses to automatic ones.

Table 10: Performance degradation when using automatic parses instead of gold-standard ones.

Classes	Task	Automatic			
		P (%)	R (%)	$F_{\beta=1}$	A (%)
ALL ARGUMENTS	Identification	88.8	80.2	84.3	
	Classification	-	-	-	87.3
	Identification + Classification	81.8	73.5	76.9	

3.9 System Analysis

Feature Salience

In analyzing the performance of the system, it is useful to estimate the relative contribution of the various feature sets used. Table 11 shows the argument classification accuracies for combinations of features on the "gold-standard" training and test set for all PropBank arguments.

Table 11: Performance of various feature combinations on the task of argument classification.

Features	Accuracy (%)	
$P(argument \mid All)$	88.92	
$P(argument \mid All\ except\ Partial\text{-}Path)$	88.58	$(\chi^2; p = 0.54)$
$P(argument \mid All\ except\ Path)$	86.61	$(\chi^2; p < 0.01)$
$P(argument \mid All\ except\ Predicate)$	83.18	
$P(argument \mid All\ except\ Head\ Word)$	83.03	
$P(argument \mid All\ except\ Head\ Word,\ Head\text{-}POS)$	82.03	
$P(argument \mid All\ except\ Predicate,\ Cluster)$	80.14	
$P(argument \mid Path,\ Predicate)$	74.40	
$P(argument \mid Path,\ Phrase\ Type)$	47.17	
$P(argument \mid Head\ Word)$	37.67	
$P(argument \mid Path)$	28.02	

Table 12: Performance of various feature combinations on the task of argument identification

Features	P (%)	R (%)	$F_{\beta=1}$
$P(is\ argument \mid All)$	94.20	89.70	91.90
$P(is\ argument \mid All\ except\ Predicate,\ Cluster)$	94.34	88.96	91.57
$P(is\ argument \mid All\ except\ Head\ Word$	92.03	87.40	89.66
$P(is\ argument \mid All\ except\ Head\ Word,\ Path,\ Predicate,\ Cluster)$	53.51	05.59	10.12

In the upper part of Table 11 we see the degradation in performance by leaving out one feature at a time. The features are arranged in the order of increasing salience. Removing all predicate 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 12 shows the feature salience 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 prohibitively slow. This indicates that the SVM is having a very hard time separating the classes from each other. Thus for the task of argument identification, the path feature is the most salient. Following that is the head word feature. Just these two features together contribute most to the accuracy of the argument identification task.

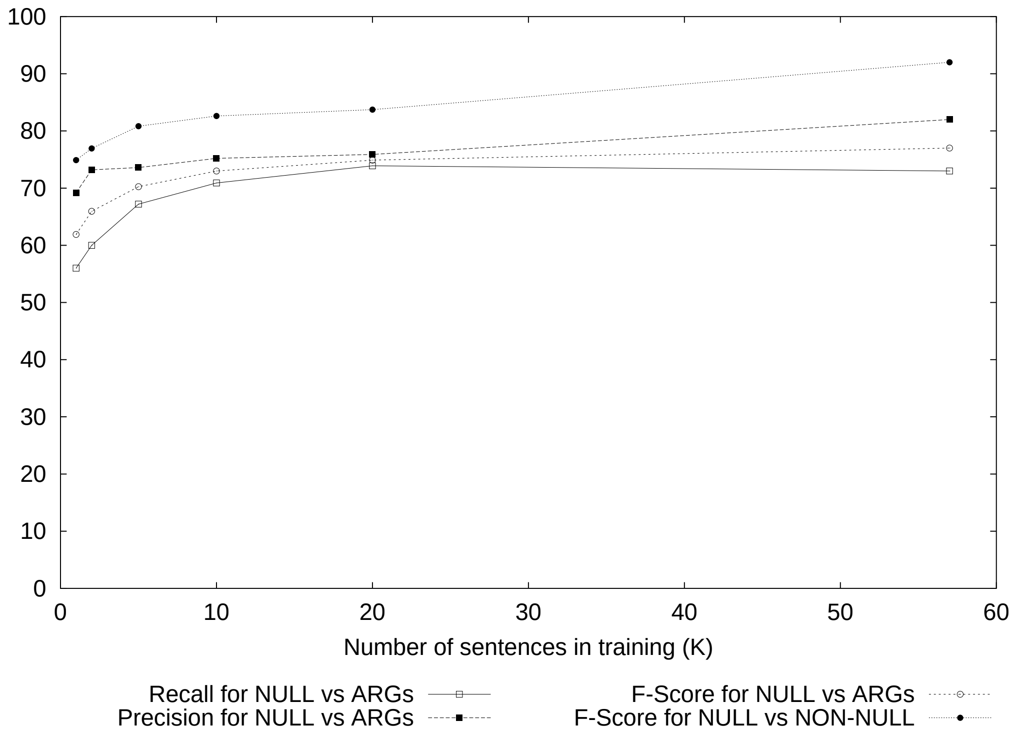


Figure 6: Learning Curve for the task of identifying and labeling arguments using automatic parses.

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 address this issue, we trained the classifier on varying amounts of training data. The resulting plots are shown in Figure 6. We approximate the curves by plotting the accuracies at six data sizes. The first curve from the top indicates the change in $F_{\beta=1}$ score on the task of argument identification alone. The second curve indicates the $F_{\beta=1}$ score on the task of argument identification and classification. Interestingly both the curves run almost parallel to each other indicating that there is constant loss due to classification errors throughout the data range. We also plotted the recall values for the combined task of identification and classification to show the fact that most of the trend in the $F_{\beta=1}$ score curve is contributed by the recall, whereas there is very little increase in precision with increasing data.

Performance Tuning

The expectations of a semantic parser 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 parser, we can tune its performance accordingly. Table 13 shows the achievable increase in precision with some drop in recall for the task of identifying and classifying semantic arguments using the "gold-standard" parses. The arguments that get assigned probability below the confidence threshold are assigned a null label.

Table 13: Precision/Recall table for the combined task of argument identification and labeling using gold-standard parses.

Confidence Threshold	Precision (%)	Recall (%)
0.00	86	81
0.10	86	80
0.25	88	77
0.50	90	71
0.75	94	58
0.90	97	35
0.95	97	16

3.10 Comparison with Similar Systems

We evaluated our system in a number of ways. First, we compare it against 4 other shallow parsers in the literature. In comparing systems, results are reported for the three types of tasks mentioned earlier.

The Gildea and Palmer (G&P) System

Gildea and Palmer (2002) use the same features used by Gildea and Jurafsky (2002), which are also our initial set of features. Two of the features – Head Word and Path – are found to be the most discriminative for argument identification. Maximum likelihood probabilities that the constituent is an argument, based on these two features – $P(is\_argument|Path, Predicate)$ and $P(is\_argument|Head, Predicate)$, are interpolated to calculate the probability that the constituent under consideration represents an argument.

In the second step, the constituents that are retained are assigned a normalized probability that is calculated by interpolating distributions conditioned on various sets of features using a backoff lattice. The most probable argument is selected. They report results on the December 2001 release of PropBank.

The Surdeanu *et al.* System

Surdeanu et al. (2003) report results for a system that uses the same features as Gildea and Jurafsky (2002) (Surdeanu System I). They then report performance gains by adding additional features (Surdeanu System II) All the feature are combined with a decision tree classifier – C5 (Quinlan, 1986; Quinlan). Using the built-in boosting capabilities of this classifier show a slight improvement on their baseline performance. They use the July 2002 release of PropBank.

The additional features used by Surdeanu System II are:

- **Content Word** – This is a lexicalized feature that selects the most informative word from a phrase using some heuristics.
- **Part of Speech of the Content Word**
- **Named Entity class of the Content Word**
- **Boolean Named Entity Flags** – The value of these features are set to true or false depending on whether the corresponding named entity is present in the phrase.
- **Phrasal verb collocations** – This feature comprises frequency statistics related to the verb and the immediately following preposition.

The Gildea and Hockenmaier (G&H) System

This system is similar to the Gildea and Palmer (2002) system, but uses features extracted from a CCG grammar instead of a phrase structure grammar. The features used are:

- **Path** – A CCG version of the path was derived.
- **Phrase type**
- **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, which contains approximately 20% more data, so we cannot directly compare to their system. However, since 80% of the data is shared by the two system, and we use the same test set, a comparison is valuable. We will refer to this as "G&H System I".

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 feature "Path" that we saw earlier, but derived from a TAG rather than from a CFG.
- **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.

We compare results to the system based on surface syntactic features (C&R System I), and the one that uses deep syntactic features (C&R System II). Unlike all the other systems we discuss, they also report results on core arguments (ARG0-5). Furthermore, owing to the absence of some semantic annotations in the Penn TreeBank, they could extract features for about 87% of the test set. Their numbers are not therefore directly comparable to ours, but since 87% of the test sets are overlapping, a rough comparison is still valuable.

3.10.1 Comparing Classifiers

Since two systems, in addition to ours, report results using the same set of features on the same data, we can directly assess the influence of the classifiers. Gildea and Palmer (2002) system estimates the posterior probabilities using several different feature sets and interpolate the estimates, while Surdeanu et al. (2003) use a decision tree classifier. Table 14 shows a comparison between the three systems for the task of argument classification. Keeping the same features and changing only the classifier produces a 8% absolute increase in performance on the same test set.

Table 14: Argument classification using same features but different classifiers

Overlaps	Accuracy (%)
SVM	87
Decision Tree (Surdeanu et al., 2003)	79
Gildea and Palmer (2002)	77

3.10.2 Argument Identification

Table 15 compares results on the argument identification task. As in the argument classification task, we see that the SVM system performs better than "Surdeanu System I" using the same set of features. It also outperforms the "Surdeanu System II", which includes some more informative features.

Table 15: Argument identification

Classes	System	Gold		
		P (%)	R (%)	$F_{\beta=1}$
All Arguments	SVM	94.2	89.7	91.9
	Surdeanu System II	-	-	89
	Surdeanu System I	85	84	85

3.10.3 Argument Classification

Table 16 compares the argument classification accuracies of the systems.

Table 16: Argument classification

Classes	System	Gold	Automatic
		Accuracy (%)	Accuracy (%)
ALL ARGUMENTS	SVM	88.6	87.3
	Surdeanu System II	84	-
	G&P	77	74
CORE ARGUMENTS	SVM	93.9	90
	C&R System II	93.5	-
	C&R System I	92.4	-

3.10.4 Argument Identification and Classification

Table 17 shows the results for combined argument identification and classification task.

Table 17: Argument identification and classification

Classes	System	Gold			Automatic		
		P (%)	R (%)	$F_{\beta=1}$	P (%)	R (%)	$F_{\beta=1}$
ALL ARGUMENTS	SVM (two-pass soft-prune)	86	82	83	82	73	77
	G&H System I	76	68	72	71	63	67
	G&P	71	64	67	58	50	54
CORE ARGUMENTS	SVM (two-pass soft-prune)	89	85	87	85	77	81
	G&H System I	82	79	80	76	73	75
	C&R System II	-	-	-	65	75	70

4 Word-by-Word Semantic Parsing

4.1 Problem Description

In the Constituent-by-Constituent (C-by-C) classification approach the candidate chunks are provided by the full syntactic parse of a sentence. So the segmentation process is performed separately from the classification task. In this method the classification is done on constituents. It is also possible to classify at the word-level. In Ramshaw and Marcus (1995), chunking was proposed as a tagging task. Here, each word in a sentence is labeled with a tag; I means that the word is inside a chunk, O means that the word is outside a chunk, and B means that the word is the beginning of a chunk. Using a variant of IOB representation, known as IOB2 (Sang and Veenstra, 1999), the words in the semantic chunks for the following example can be tagged as shown:

[$_{\textbf{ARG1}}$I] 'm [$_{\textbf{predicate}}$inspired] by [$_{\textbf{ARG0}}$the mood of the people].

[$_{\textbf{B-ARG1}}$ I] [$_{\textbf{O}}$ 'm] [$_{\textbf{O}}$ inspired] [$_{\textbf{O}}$ by] [$_{\textbf{B-ARG0}}$ the][$_{\textbf{I-ARG0}}$ mood] [$_{\textbf{I-ARG0}}$ of] [$_{\textbf{I-ARG0}}$ the] [$_{\textbf{I-ARG0}}$ people]

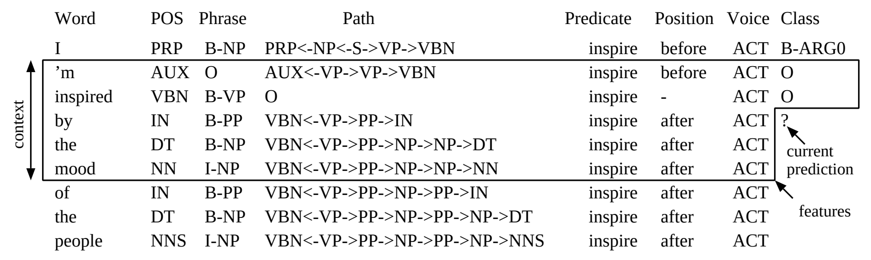
In this scheme, N arguments would result in $2*N+1$ classes. Experiments using this method have been reported in Hacioglu and Ward (2003); Pradhan et al. (2003); Hacioglu et al. (2003); Pradhan et al.

4.2 Features

The features are very similar to the features used in the C-by-C system, except the features are derived for each word, not for each constituent. The features are

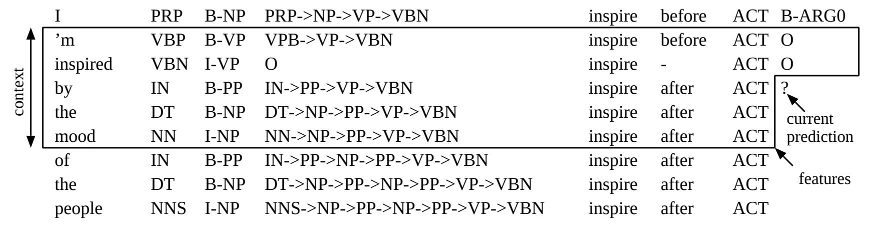
- **Predicate**: The verb in focus.
- **Part of Speech**: The part-of-speech category of the word.
- **Phrase Position**: The position of the word in a phrase based on IOB representation (e.g. B-NP, I-NP, O etc.)
- **Word Position**: The position of the word with respect to the predicate. It has two values as "before" and "after".
- **Voice**: Indicates whether the sentence is in active or passive form with respect to the predicate.
- **Path**: This is same as the path feature used in the C-by-C system, except the path is relative to the word and not the constituent.

For each word to be tagged, a set of features is created from a fixed size context that surrounds the word. A 5-word sliding window centered at the current word defines the set of features. In addition to the above features, we also use semantic IOB tags that have already been assigned to the words that precede the word in focus and appear in the context. Figure 7 clarifies the notion of context and illustrates the features used. It should be noted that some of the features attached to the words are sentence-level features.



Word	POS	Phrase	Path	Predicate	Position	Voice	Class
I	PRP	B-NP	PRP<-NP<-S->VP->VBN	inspire	before	ACT	B-ARG0
’m	AUX	O	AUX<-VP->VP->VBN	inspire	before	ACT	O
inspired	VBN	B-VP	O	inspire	-	ACT	O
by	IN	B-PP	VBN<-VP->PP->IN	inspire	after	ACT	?
the	DT	B-NP	VBN<-VP->PP->NP->NP->DT	inspire	after	ACT	
mood	NN	I-NP	VBN<-VP->PP->NP->NP->NN	inspire	after	ACT	
of	IN	B-PP	VBN<-VP->PP->NP->PP->IN	inspire	after	ACT	
the	DT	B-NP	VBN<-VP->PP->NP->PP->NP->DT	inspire	after	ACT	
people	NNS	I-NP	VBN<-VP->PP->NP->PP->NP->NNS	inspire	after	ACT	

Figure 7: Illustration of the context and the features used to classify a word – Chunker-1



I	PRP	B-NP	PRP->NP->VP->VBN	inspire	before	ACT	B-ARG0
’m	VBP	B-VP	VPB->VP->VBN	inspire	before	ACT	O
inspired	VBN	I-VP	O	inspire	-	ACT	O
by	IN	B-PP	IN->PP->VP->VBN	inspire	after	ACT	?
the	DT	B-NP	DT->NP->PP->VP->VBN	inspire	after	ACT	
mood	NN	I-NP	NN->NP->PP->VP->VBN	inspire	after	ACT	
of	IN	B-PP	IN->PP->NP->PP->VP->VBN	inspire	after	ACT	
the	DT	B-NP	DT->NP->PP->NP->PP->VP->VBN	inspire	after	ACT	
people	NNS	I-NP	NNS->NP->PP->NP->PP->VP->VBN	inspire	after	ACT	

Figure 8: Illustration of the context and the features used to classify a word – Chunker-2

4.3 System Implementation

As in the C-by-C classification framework we have used SVMs in the One *vs* All setup. All SVM classifiers are realized using the TinySVM with the polynomial kernel of degree 2 and the general purpose SVM based chunker YamCha. Figures 9 and 10 illustrate two implementations for feature extraction. As in the baseline system, the first system extracts features from a full syntactic parser. Even though classification is done on a word-by-word basis, features are extracted from a full parse. This system is referred to as W-by-W Chunker-1. In the second system, the full-syntactic parser is replaced by a part of speech tagger followed by a syntactic phrase chunker (YamCha). This system is referred to as W-by-W Chunker-2. As mentioned earlier, the motivations for using a chunking parser instead of a full syntactic parser are: i) full parsing is computationally expensive than chunking, ii) chunkers can be developed more easily for new languages and genre of text. The features created for these two systems are shown in Figure 8 for the example exhibited in Figure 7. There are slight differences in POS and phrase position features and major differences in the path features. The path feature for W-by-W Chunker 2 is uni-directional (from the word in focus to the predicate) and created using the flat structure:

[**NP** (**PRP** I)] [**VP** (**VBP** ’m) (**VBN** inspired)] [**PP** (**IN** by)] [**NP** (**DT** the) (**NN** mood)] [**PP** (**IN** of)] [**NP** (**DT** the) (**NNS** people)]

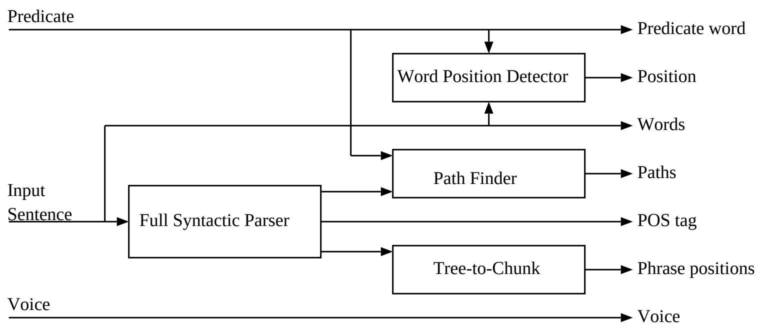


Figure 9: Chunker-1 architecture

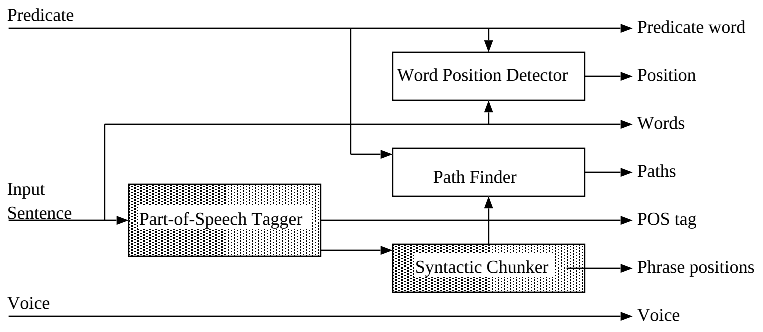


Figure 10: Chunker-2 architecture

We are experimenting with both implementations to determine trade-offs between coverage, efficiency and accuracy. We present some preliminary results in the next section that compare the accuracy of Chunker-2 to Chunker-1, and also we compare their performance to the C-by-C parser described in Section 3.

4.4 Experimental Results

Since training SVM on the entire PropBank corpus takes a long time, these experiments were carried out using an SVM trained on 10,000 randomly selected sentences from the training data. As always, Section-23 was used as the test set. We used the Charniak parser to obtain syntactic parse trees. The results are shown in Table 18.

When features are derived from a flat chunked parse, instead of a syntactic parse tree, we observed a significant drop in performance. The major difference between these systems is the derivation of the path feature. To further illustrate the effect, we ran both systems without using the path feature. These results are shown in Table 19. As can be seen the performance of these systems is very similar, indicating the salience of the path feature. These experiments have also shown that the performance obtained by the W-by-W paradigm fell

Table 18: Comparison of C-by-C and W-by-W classifiers

System	P	R	$F_{\beta=1}$
C-by-C	80.6	67.1	73.2
Chunker-1, W-by-W	70.7	60.5	65.2
Chunker-2, W-by-W	66.2	54.9	60.0

Table 19: Comparison W-by-W classifiers without the path feature

System	P	R	$F_{\beta=1}$
Chunker-1, W-by-W	60.4	51.4	55.6
Chunker-2, W-by-W	59.8	50.8	54.9

short of that obtained by the C-by-C paradigm. We think that an important step towards bridging this gap would be to adopt a two pass approach in the W-by-W paradigm, analogous in the C-by-C paradigm. That is, first chunk the sentence into arguments (non-Nulls) and non-arguments(Nulls), and then label the non-Nulls with the respective arguments. We are currently working along those directions to improve the performance.

5 Generalization to a New Domain

Thus far, all experiments have been tested on the same genre of data that they have been trained on. In order to see how well the features transfer to a new domain, we used both the W-by-W and the C-by-C classifiers trained on PropBank to test data drawn from the AQUAINT corpus (LDC, 2002). We annotated 400 sentences from the AQUAINT corpus with PropBank arguments. This is a collection of text from the New York Times Inc., Associated Press Inc., and Xinhua News Service (PropBank by comparison is drawn from Wall Street Journal). The results of the C-by-C classifier are shown in Table 20.

Table 20: Performance of the C-by-C classifier on the AQUAINT test set

	Task	P (%)	R (%)	$F_{\beta=1}$
All Arguments	Identification	74	70	72
All Arguments	Identification + Classification	64	60	62
Core Arguments	Identification + Classification	76	66	71

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 73% 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. All the other features are less likely to have been a source of the problem since they can take small enough values that the amount of training data is sufficient to estimate their statistics.

Table 21: Feature Coverage on PropBank test set using parser trained on PropBank training set.

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

Table 22: Coverage of features on AQUAINT test set using parser trained on PropBank training set.

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

Table 21 shows the coverage for features on the "gold-standard" 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 22 shows coverage for the same features on the AQUAINT test set. We believe that the drop in coverage of the more predictive feature combinations explains part of the drop in performance. We also ran the Word-by-Word chunking algorithms on the AQUAINT test set. The results are shown in Table 23. This would compare to the Table 18. We also observed a degradation in the $F_{\beta=1}$ score for this condition, from 65.2% to 52.2% for Chunker-1 and 60.0% to 45.7% for

Chunker-2.

Table 23: Identification and Classification performance of the chunking systems on the AQUAINT test set

Classes	System	P (%)	R (%)	$F_{\beta=1}$
ALL	Chunker-1, W-by-W	54.2	50.4	52.2
ARGUMENTS	Chunker-2, W-by-W	49.5	42.4	45.7

6 Conclusions

We have described an algorithm which significantly improves the state-of-the-art in shallow semantic parsing. Like previous work, our parser is based on a supervised machine learning approach. It achieves the best published performance on the task of identifying and labeling semantic arguments in PropBank. Key aspects of our results include significant (8% absolute) improvement via an SVM classifier, improvement from new features, and a series of analytic experiments on the contributions of the features.

For example, we found that path is the most salient feature for argument identification, but is difficult to generalize. Only the partial-path generalization seemed to have any benefit. For the task of classification, head word and predicate were the most salient features, but likewise, may be difficult to estimate because of data sparsity. Adding features that are generalizations of the more specific features also seemed to help. These features were named entities, head word part of speech and verb clusters.

We also report on a series of experiments using a completely different architecture which does not require a syntactic parser. This architecture performed significantly worse than our parser based architecture, but may nonetheless be useful in new languages or genres in which statistical syntactic parsers are not available.

7 Acknowledgments

We would like to thank Ralph Weischedel and Scott Miller of BBN Inc. for letting us use their named entity tagger – IdentiFinder; Daniel Gildea for providing the source for his semantic role tagger and for helpful discussions; Martha Palmer for providing us with the PropBank data and useful advice; and Steven Bethard and other members of CSLR for help and suggestions.

References

Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. In *Proc. 17th International Conf. on Machine Learning*, pages 9–16. Morgan Kaufmann, San Francisco, CA, 2000.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley FrameNet project. In *COLING/ACL-98*, pages 86–90, Montreal, 1998. ACL.

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.

Eugene Chaniak. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Conference of the Association for Computational Linguistics (ACL-01)*, Toulouse, France, 2001.

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 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.

Michael Fleischman and Eduard Hovy. A maximum entropy approach to framenet tagging. In *Proceedings of the Human Language Technology Conference*, Edmonton, Canada, 2003.

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 and Daniel Jurafsky. Automatic labeling of semantic roles. In *Proceedings of the 38th Annual Conference of the Association for Computational Linguistics (ACL-00)*, pages 512–520, Hong Kong, October 2000.

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.

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 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.

Marti Hearst. Untangling text data mining. In *Proceedings of the 37th Annual Meeting of the ACL*, pages 3–10, College Park, Maryland, 1999.

Thomas Hofmann and Jan Puzicha. Statistical models for co-occurrence data. Memo, Massachussetts Institute of Technology Artificial Intelligence Laboratory, February 1998.

Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning (ECML)*, 1998.

Paul Kingsbury, Martha Palmer, and Mitch Marcus. Adding semantic annotation to the Penn Treebank. In *Proceedings of the Human Language Technology Conference*, San Diego, CA, 2002.

Ulrich H G Krebel. Pairwise classification and support vector machines. In *Advances in Kernel Methods*, 1999.

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.

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 COLING-ACL*, Montreal, Canada, 1998.

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.

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, 1994.

John Platt. Probabilities for support vector machines. In A. Smola, P. Bartlett, B. Scolkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers.* MIT press, Cambridge, MA, 2000.

Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James Martin, and Dan Jurafsky. Experiments with statistical semantic role parsing number =. Technical report.

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, 2003.

J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

Ross Quinlan. Data Mining Tools See5 and C5.0, `http://www.rulequest.com./see-info.html`, 2003.

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.

Erik F. Tjong Kim Sang and Jorn Veenstra. Representing text chunks. In *Proc. of EACL*, pages 173–179, 1999.

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.

Cynthia A. Thompson, Roger Levy, and Christopher D. Manning. A generative model for semantic role labeling. In *Proceedings of the European Conference on Machine Learning (ECML)*, 2003.

Sean Wallis and Gerald Nelson. Knowledge discovery in grammatically analysed corpora. *Data Mining and Knowledge Discovery*, 5(4):305–335, 2001.