Shallow Semantic Parsing using Support Vector Machines*

Sameer Pradhan, Wayne Ward, Kadri Hacioglu,
James H. Martin and Dan Jurafsky†
Center for Spoken Language Research,
University of Colorado, Boulder, CO 80303
{spradhan,whw,hacioglu,martin}@cslr.colorado.edu

Abstract

In this paper, we propose a machine learning algorithm for shallow semantic 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 an improvement in performance over earlier classifiers. We show performance improvements through a number of new features and measure their ability to generalize to a new test set drawn from the AQUAINT corpus.

1 Introduction

Automatic, accurate and wide-coverage techniques that can annotate naturally occurring text with semantic argument structure can play a key role in NLP applications such as Information Extraction, Question Answering and Summarization. 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. 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.

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, 2002); Blaheta and Charniak (2000); Gildea and Palmer (2002); Surdeanu et al. (2003); Gildea and Hockenmaier (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) (G&J) 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 to the existing feature set. We evaluate results using both hand-corrected TreeBank syntactic parses, and actual parses from the Charniak parser.

2 Semantic Annotation and Corpora

We will be reporting on results using PropBank[1] (Kingsbury et al., 2002), a 300k-word corpus in which predicate argument relations are marked for part of the verbs in the Wall Street Journal (WSJ) part of the Penn TreeBank (Marcus et al., 1994). The arguments of a verb are labeled 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 attempts 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. Figure 1 shows the syntax tree representation along with the argument labels for an example structure extracted from the PropBank corpus.

Most of the experiments in this paper, unless specified otherwise, are performed on the July 2002 release of PropBank. A larger, cleaner, completely adjudicated version of PropBank was made available in Feb 2004. We will also report some final best performance numbers on this corpus. PropBank was constructed by assigning semantic arguments to constituents of the hand-corrected TreeBank parses. The data comprise several sections of

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

† Currently at Stanford University, Stanford, CA 94305

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

the WSJ, and we follow the standard convention of using Section-23 data as the test set. Section-02 to Section-21 were used for training. In the July 2002 release, 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. The Feb 2004 release training set comprises about 85,000 sentences instantiating about 250,000 arguments and the test set comprises 5,000 sentences instantiating about 12,000 arguments.

[$_{\mathbf{ARG0}}$ He] [$_{\mathbf{predicate}}$ talked] for [$_{\mathbf{ARGM-TMP}}$ about 20 minutes].

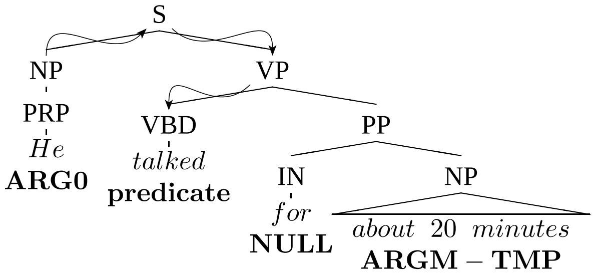

Figure 1: Syntax tree for a sentence illustrating the PropBank tags.

3 Problem Description

The problem of shallow semantic parsing can be viewed as three different tasks.

Argument Identification – This is the process of identifying parsed constituents in the sentence that represent semantic arguments of a given predicate.

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

Argument Identification and Classification – A combination of the above two tasks.

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 1, the node IN that encompasses "for" is a NULL node because it does not correspond to a semantic argument. The node NP that encompasses "about 20 minutes" is a NON-NULL node, since it does correspond to a semantic argument – ARGM-TMP.

4 Baseline Features

Our baseline system uses the same set of features introduced by G&J. 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.
- **Path –** The syntactic path through the parse tree from the parse constituent to the predicate being classified. For example, in Figure 1, the path from ARG0 – "He" to the predicate *talked*, is represented with the string NP↑S↓VP↓VBD. ↑ and ↓ represent upward and downward movement in the tree respectively.
- **Phrase Type –** This is the syntactic category (NP, PP, S, etc.) of the phrase/constituent 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.
- **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).
- **Sub-categorization –** This is the phrase structure rule expanding the predicate's parent node in the parse tree. For example, in Figure 1, the sub-categorization for the predicate *talked* is VP→VBD-PP.

5 Classifier and Implementation

We formulate the parsing problem as a multi-class classification problem and use a Support Vector Machine (SVM) classifier (Hacioglu et al., 2003; Pradhan et al, 2003). Since SVMs are binary classifiers, we have to convert the multi-class problem into a number of binary-class problems. We use the ONE *vs* ALL (OVA) formalism, which involves training n binary classifiers for a n-class problem.

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, while maintaining the same accuracy:

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).
2. The remaining training data is used to train OVA classifiers, one of which is the NULL-NON-NULL classifier.

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. All the nodes are classified directly as NULL or one of the arguments using the classifier trained in step 2 above. We observe no significant performance improvement even if we filter the most likely NULL nodes in a first pass.

For our experiments, we used TinySVM[2] along with YamCha[3] (Kudo and Matsumoto, 2000) (Kudo and Matsumoto, 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.

6 Baseline System Performance

Table 1 shows the baseline performance numbers on the three tasks mentioned earlier; these results are based on syntactic features computed from hand-corrected TreeBank (hence LDC hand-corrected) parses.

For the argument identification and the combined identification and classification tasks, we report the precision (P), recall (R) and the F_1[4] 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.

Classes	Task	P (%)	R (%)	F_1	A (%)
ALL ARGS	Id. Classification Id. + Classification	90.9 - 83.3	89.8 - 78.5	90.4 - 80.8	87.9
CORE ARGS	Id. Classification Id. + Classification	94.7 - 88.4	90.1 - 84.1	92.3 - 86.2	91.4

Table 1: Baseline performance on all three tasks using hand-corrected parses.

7 System Improvements

7.1 Disallowing Overlaps

The system as described above might label two constituents NON-NULL even if they overlap in words. This is a problem since overlapping arguments are not allowed in PropBank. Among the overlapping constituents we retain the one for which the SVM has the highest confidence, and label the others NULL. The probabilities obtained by applying the sigmoid function to the raw SVM scores are used as the measure of confidence. Table 2 shows the performance of the parser on the task of identifying and labeling semantic arguments using the hand-corrected parses. On all the system improvements, we perform a χ^2 test of significance at $p = 0.05$, and all the significant improvements are marked with an *. In this system, the overlap-removal decisions are taken independently of each other.

	P (%)	R (%)	F_1
Baseline	83.3	78.5	80.8
No Overlaps	85.4	78.1	*81.6

Table 2: Improvements on the task of argument identification and classification after disallowing overlapping constituents.

7.2 New Features

We tested several new features. Two were obtained from the literature – named entities in constituents and head word part of speech. Other are novel features.

1. **Named Entities in Constituents** – Following Surdeanu et al. (2003), 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.

2. **Head Word POS** – Surdeanu et al. (2003) showed that using the part of speech (POS) of the head word gave a significant performance boost to their system. Following that, we experimented with the addition of this feature to our system.

3. **Verb Clustering** – 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 verbs were clustered into 64 classes using the probabilistic co-occurrence model of Hofmann and Puzicha (1998). The clustering algorithm uses a database of verb-direct-object relations extracted by Lin (1998). We then use the verb class of the current predicate as a feature.

4. **Partial Path** – For the argument identification task, path is the most salient feature. However, it is also the most data sparse feature. To overcome this problem, we tried generalizing the path by adding a new feature that contains only the part of the path from the constituent to the lowest common ancestor of the predicate and the constituent, which we call "Partial-Path".

5. **Verb Sense Information** – The arguments that a predicate can take depend on the word 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] http://cl.aist-nara.ac.jp/~talus-Au/software/TinySVM/

[3] http://cl.aist-nara.ac.jp/~taku-Au/software/yamcha/

[4] $F_1 = \frac{2PR}{P+R}$

illustrates the argument sets for the predicate *talk*. 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.

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

Table 3: Argument labels associated with the two senses of predicate *talk* in PropBank corpus.

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

6. **Head Word 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, which are always prepositions, are not very discriminative, eg., “in the city”, “in a few minutes”, both share the same head word “in” and neither contain a named entity, but the former is ARGM-LOC, whereas the latter is ARGM-TMP. Therefore, we tried replacing the head word of a prepositional phrase, with that of the first noun phrase inside the prepositional phrase. We retained the preposition information by appending it to the phrase type, eg., “PP-in” instead of “PP”.

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.

8. **Ordinal constituent position –** In order to avoid false positives of the type 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.

9. **Constituent tree distance –** This is a finer way of specifying the already present position feature.

10. **Constituent relative features –** These are 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 in focus. These were added on the intuition that encoding the tree context this way might add robustness and improve generalization.

11. **Temporal cue words –** There are several temporal cue words that are not captured by the named entity tagger and were considered for addition as a binary feature indicating their presence.

12. **Dynamic class context –** In the task of argument classification, these are dynamic features that represent the hypotheses of at most previous two nodes belonging to the same tree as the node being classified.

8 Feature Performance

Table 4 shows the effect each feature has on the argument classification and argument identification tasks, when added individually to the baseline. Addition of named entities improves the F_1 score for adjunctive arguments ARGM-LOC from 59% to *68% and ARGM-TMP from 78.8% to *83.4%. But, since these arguments are small in number compared to the core arguments, the overall accuracy does not show a significant improvement. We found that adding this feature to the NULL *vs* NON-NULL classifier degraded its performance. It 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. Apparently, 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.

Features	Class Acc.	ARGUMENT ID		
		P	R	F_1
Baseline	87.9	93.7	88.9	91.3
+ Named entities	**88.1**	-	-	-
+ 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 4: Effect of each feature on the argument identification and classification tasks when added to the baseline system.

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 any improvement. We also tried generalizing the path feature by i) compressing sequences of identical labels, and ii) removing the direction in the path, but none showed any improvement on the baseline.

8.1 Argument Sequence Information

In order to improve the performance of their statistical argument tagger, G&J used the fact that a predicate is likely to instantiate a certain set of arguments. 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 G&J, we can also estimate the probability of argument sets not seen in the training data. We first convert the raw SVM scores to probabilities using a sigmoid function. 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	90.0	86.1	88.0
Common predicate	90.8	86.3	88.5
Specific predicate lemma	90.5	87.4	*88.9

Table 5: Improvements on the task of argument identification and tagging 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 Argument accuracy on the combined task of identifying and assigning semantic arguments, given hand-corrected parses, whereas the accuracy of the Adjunctive Arguments slightly deteriorated. This seems to be logical considering the fact that the Adjunctive Arguments are not linguistically constrained in any way as to their position in the sequence of arguments, or 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 5.

9 Best System Performance

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 4, 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 4. Table 6 shows the performance of this system.

Classes	Task	Hand-corrected parses			
		P (%)	R (%)	F_1	A (%)
All Args	Id.	95.2	92.5	93.8	91.0
	Classification	-	-	-	
	Id. + Classification	88.9	84.6	86.7	
Core Args	Id.	96.2	93.0	94.6	93.9
	Classification	-	-	-	
	Id. + Classification	90.5	87.4	88.9	

Table 6: Best system performance on all tasks using hand-corrected parses.

10 Using Automatic Parses

Thus far, we have reported results using hand-corrected 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. We lemmatized the predicate using the XTAG morphology database[5] (Daniel et al., 1992). Table 7 shows the performance degradation when automatically generated parses are used.

11 Using Latest PropBank Data

Owing to the Feb 2004 release of much more and completely adjudicated PropBank data, we have a chance to

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

Classes	Task	Automatic parses			
		P (%)	R (%)	F_1	A (%)
All Args	Id.	89.3	82.9	86.0	
	Classification	-	-	-	90.0
	Id. + Classification	84.0	75.3	79.4	
Core Args	Id.	92.0	83.3	87.4	
	Classification	-	-	-	90.5
	Id. + Classification	86.4	78.4	82.2	

Table 7: Performance degradation when using automatic parses instead of hand-corrected ones.

report our performance numbers on this data set. Table 8 shows the same information as in previous Tables 6 and 7, but generated using the new data. Owing to time limitations, we could not get the results on the argument identification task and the combined argument identification and classification task using automatic parses.

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	Classification	-	-	-	90.1

Table 8: Best system performance on all tasks using hand-corrected parses using the latest PropBank data.

12 Feature Analysis

In analyzing the performance of the system, it is useful to estimate the relative contribution of the various feature sets used. Table 9 shows the argument classification accuracies for combinations of features on the training and test data, using hand-corrected parses, for all PropBank arguments.

Features	Accuracy (%)
All	91.0
All except Path	90.8
All except Phrase Type	90.8
All except HW and HW-POS	90.7
All except All Phrases	*83.6
All except Predicate	*82.4
All except HW and FW and LW-POS	*75.1
Path, Predicate	74.4
Path, Phrase Type	47.2
Head Word	37.7
Path	28.0

Table 9: Performance of various feature combinations on the task of argument classification.

In the upper part of Table 9 we see the degradation in performance by leaving out one feature or a feature family at a time. After the addition of all the new features, it is the case that removal of no individual feature except predicate degrades the classification performance significantly, as there are some other features that provide complimentary information. However, removal of predicate information hurts performance significantly, so does the removal of a family of features, eg., all phrase types, or the head word (HW), first word (FW) and last word (LW) information. The lower part of the table shows the performance of some feature combinations by themselves.

Table 10 shows the feature salience on the task of argument identification. One important observation we can make here is that the path feature is the most salient feature in the task of argument identification, whereas it is the least salient in the task of argument classification. We could not provide the numbers for argument identification performance upon removal of the path feature since that made the SVM training prohibitively slow, indicating that the SVM had a very hard time separating the Null class from the non-Null class.

Features	P (%)	R (%)	F_1
All	95.2	92.5	93.8
All except HW	95.1	92.3	93.7
All except Predicate	94.5	91.9	93.2

Table 10: Performance of various feature combinations on the task of argument identification

13 Comparing Performance with Other Systems

We compare our system against 4 other shallow semantic parsers in the literature. In comparing systems, results are reported for all the three types of tasks mentioned earlier.

13.1 Description of the Systems

The Gildea and Palmer (G&P) System.

The Gildea and Palmer (2002) system uses the same features and the same classification mechanism used by G&J. These results are reported on the December 2001 release of PropBank.

The Surdeanu et al. System.

Surdeanu et al. (2003) report results on two systems using a decision tree classifier. One that uses exactly the same features as the G&J system. We call this “Surdeanu System I.” They then show improved performance of another system – “Surdeanu System II,” which uses some additional features. These results are are reported on the July 2002 release of PropBank.

The Gildea and Hockenmaier (G&H) System

The Gildea and Hockenmaier (2003) system uses features extracted from Combinatory Categorial Grammar (CCG) corresponding to the features that were used by G&J and G&P systems. CCG is a form of dependency grammar and is hoped to capture long distance relationships better than a phrase structure grammar. The features are combined using the same algorithm as in G&J

and G&P. They use a slightly newer – November 2002 release of PropBank. We will refer to this as "G&H System I".

The Chen and Rambow (C&R) System

Chen and Rambow report on two different systems, also using a decision tree classifier. The first "C&R System I" uses surface syntactic features much like the G&P system. The second "C&R System II" uses additional syntactic and semantic representations that are extracted from a Tree Adjoining Grammar (TAG) – another grammar formalism that better captures the syntactic properties of natural languages.

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

Table 11: Argument classification using same features but different classifiers.

13.2 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. G&P 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 11 shows a comparison between the three systems for the task of argument classification.

13.3 Argument Identification (Null vs non-Null)

Table 12 compares the results of the task of identifying the parse constituents that represent semantic arguments. As expected, the performance degrades considerably when we extract features from an automatic parse as opposed to a hand-corrected 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.

Classes	System	Hand			Automatic		
		P	R	F_1	P	R	F_1
All Args	SVM	95	92	94	89	83	86
	Surdeanu System II	-	-	89	-	-	-
	Surdeanu System I	85	84	85	-	-	-

Table 12: Argument identification

13.4 Argument Classification

Table 13 compares the argument classification accuracies of various systems, and at various levels of classification granularity, and parse accuracy. It can be seen that the SVM System performs significantly better than all the other systems on all PropBank arguments.

Classes	System	Hand Accuracy	Automatic Accuracy
All Args	SVM	91	90
	G&P	77	74
	Surdeanu System II	84	-
	Surdeanu System I	79	-
Core Args	SVM	93.9	90.5
	C&R System II	93.5	-
	C&R System I	92.4	-

Table 13: Argument classification

13.5 Argument Identification and Classification

Table 14 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. SVM does a very good job of generalizing in both stages of processing.

Classes	System	Hand			Automatic		
		P	R	F_1	P	R	F_1
All Args	SVM	89	85	87	84	75	79
	G&H System I	76	68	72	71	63	67
	G&P	71	64	67	58	50	54
Core Args	SVM System	90	87	89	86	78	82
	G&H System I	82	79	80	76	73	75
	C&R System II	-	-	-	65	75	70

Table 14: Identification and classification

14 Generalization to a New Text Source

Thus far, in all experiments our unseen test data was selected from the same source as the training data. In order to see how well the features generalize to texts drawn from a similar source, we used the classifier trained on PropBank training data 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 are shown in Table 15.

	Task	P (%)	R (%)	F_1	A (%)
All Args	Id.	75.8	71.4	73.5	-
	Classification	-	-	-	83.8
	Id. + Classification	65.2	61.5	63.3	-
Core Args	Id.	88.4	74.4	80.8	-
	Classification	-	-	-	84.0
	Id. + Classification	75.2	63.3	68.7	-

Table 15: 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 84% and 75% 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 (%)
$Predicate, Path$	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 16: Feature Coverage on PropBank 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 17: Coverage of features on AQUAINT test set using parser trained on PropBank training set.

Table 16 shows the coverage for features on the hand-corrected 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 17 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.

15 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. Key aspects of our results include significant improvement via an SVM classifier, improvement from new features and a series of analytic experiments on the contributions of the features. Adding features that are generalizations of the more specific features seemed to help. These features were named entities, head word part of speech and verb clusters. We also analyzed the transferability of the features to a new text source.

We would like to thank Ralph Weischedel and Scott Miller of BBN Inc. for letting us use their named entity tagger – IdentiFinder; Martha Palmer for providing us with the PropBank data, Valerie Krugler for tagging the AQUAINT test set with PropBank arguments, and all the anonymous reviewers for their helpful comments.

References

[Bikel et al.1999] Daniel M. Bikel, Richard Schwartz, and Ralph M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*, 34:211–231.

[Blaheta and Charniak2000] Don Blaheta and Eugene Charniak. 2000. Assigning function tags to parsed text. In *NAACL*, pages 234–240.

[Chaniak2001] Eugene Chaniak. 2001. Immediate-head parsing for language models. In *ACL-01*.

[Chen and Rambow2003] John Chen and Owen Rambow. 2003. Use of deep linguistics features for the recognition and labeling of semantic arguments. *EMNLP-03*.

[Collins1999] Michael John Collins. 1999. *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia.

[Daniel et al.1992] K. Daniel, Y. Schabes, M. Zaidel, and D. Egedi. 1992. A freely available wide coverage morphological analyzer for English. In *COLING-92*.

[Fleischman and Hovy2003] Michael Fleischman and Eduard Hovy. 2003. A maximum entropy approach to framenet tagging. In *HLT-03*.

[Gildea and Hockenmaier2003] Dan Gildea and Julia Hockenmaier. 2003. Identifying semantic roles using combinatory categorial grammar. In *EMNLP-03*.

[Gildea and Jurafsky2000] Daniel Gildea and Daniel Jurafsky. 2000. Automatic labeling of semantic roles. In *ACL-00*, pages 512–520.

[Gildea and Jurafsky2002] Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

[Gildea and Palmer2002] Daniel Gildea and Martha Palmer. 2002. The necessity of syntactic parsing for predicate argument recognition. In *ACL-02*.

[Hacioglu and Ward2003] Kadri Hacioglu and Wayne Ward. 2003. Target word detection and semantic role chunking using support vector machines. In *HLT-03*.

[Hacioglu et al.2003] Kadri Hacioglu, Sameer Pradhan, Wayne Ward, James Martin, and Dan Jurafsky. 2003. Shallow semantic parsing using support vector machines. Technical Report TR-CSLR-2003-1, Center for Spoken Language Research, Boulder, Colorado.

[Hofmann and Puzicha1998] Thomas Hofmann and Jan Puzicha. 1998. Statistical models for co-occurrence data. Memo, MIT AI Laboratory.

[Kingsbury et al.2002] Paul Kingsbury, Martha Palmer, and Mitch Marcus. 2002. Adding semantic annotation to the Penn Treebank. In *HLT-02*.

[Kudo and Matsumoto2000] Taku Kudo and Yuji Matsumoto. 2000. Use of support vector learning for chunk identification. In *CoNLL-00*.

[Kudo and Matsumoto2001] Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *NAACL-01*.

[LDC2002] LDC. 2002. The AQUAINT Corpus of English News Text, Catalog no. LDC2002t31.

[Lin1998] Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *COLING-98*.

[Magerman1994] David Magerman. 1994. *Natural Language Parsing as Statistical Pattern Recognition*. Ph.D. thesis, Stanford University, CA.

[Marcus et al.1994] Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn TreeBank: Annotating predicate argument structure.

[Platt2000] John Platt. 2000. Probabilities for support vector machines. In A. Smola, P. Bartlett, B. Scolkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT press.

[Pradhan et al.2003] Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James Martin, and Dan Jurafsky. 2003. Semantic role parsing: Adding semantic structure to unstructured text. In *ICDM-03*.

[Surdeanu et al.2003] Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *ACL-03*.

[Thompson et al.2003] Cynthia A. Thompson, Roger Levy, and Christopher D. Manning. 2003. A generative model for semantic role labeling. In *ECML-03*.