Semantic Role Parsing: Adding Semantic Structure to Unstructured Text*

Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H. Martin, Daniel Jurafsky
Center for Spoken Language Research,
University of Colorado, Boulder, CO 80303

Abstract

There is a ever-growing need to add structure in the form of semantic markup to the huge amounts of unstructured text data now available. We present the technique of shallow semantic parsing, the process of assigning a simple WHO *did* WHAT *to* WHOM*, etc., structure to sentences in text, as a useful tool in achieving this goal. We formulate the semantic parsing problem as a classification problem using Support Vector Machines. Using a hand-labeled training set and a set of features drawn from earlier work together with some feature enhancements, we demonstrate a system that performs better than all other published results on shallow semantic parsing.*

1. Introduction

Automatic, accurate, wide-coverage techniques that can annotate naturally occurring text with semantic roles can facilitate the discovery of patterns of information in large text collections [15]. Shallow semantic parsing is a process for producing such a markup. In shallow semantic parsing, semantic tags are assigned to the arguments, or case roles, associated with each predicate in the sentence. This technique is used widely in Information Extraction, and is being evaluated for use in Summarization, Question Answering and Machine Translation.

We treat the problem of tagging parsed constituents as a multi-class classification problem, where the classifier is trained in a supervised manner from human-annotated data using Support Vector Machines [17]. The next section describes our training and test corpora. Rest of the paper describes our system in detail, compares it to other systems, presents some analysis, and points to future work.

2. Semantic Annotation and Corpora

There are many possible approaches to specifying the roles to be used for the markup. Two corpora are available for developing and testing semantic annotation – FrameNet[1] [1] and PropBank[2] [11]. 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 one million word corpus in which predicate argument relations are marked for every occurrence of every verb in the Wall Street Journal (WSJ) part of the Penn TreeBank [13]. The arguments of a verb are labeled sequentially from ARG0 to ARG5, where ARG0 is usually the subject of a transitive verb; ARG1, its direct object, etc. In addition to these "core arguments," additional "adjunctive arguments," for example, ARGM-LOC, for locatives, and ARGM-TMP, for temporals, are also marked. We will refer to these as ARGMs. An example PropBank style markup:

1. [$_{ARG0}$ Merrill Lynch Co.] refuses to [$_{predicate}$ perform] [$_{ARG1}$ index arbitrage trades] for [$_{ARG2}$ clients.]

All experiments in this paper are performed on the July 2002 release of PropBank. In these experiments, the test set is Section-23 of the WSJ data. Section-02 through Section-21 are used for training. The training set comprises approximately 51,000 sentences with 132,000 arguments, and the test set comprises approximately 2,700 sentences with 7,000 arguments.

3. System Architecture

The basic steps of our shallow semantic parser are similar to those outlined by Gildea & Jurafsky (G&J) [6]:

procedure $Parse(Sentence)$
- Generate a full syntactic parse for the $Sentence$
- Identify all verb $predicates$

for $predicate \in Sentence$ **do**
- Extract a set of features for each node in the tree relative to the $predicate$
- Classify each node as NULL, or as one of the PropBank arguments
- Generate Parse

end for

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

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

The features used by the classifier are:

Predicate – The predicate itself is used as a feature.

Path – The syntactic path through the parse tree from the governing predicate to the parse constituent being classified. For example, in Figure 1, the path from ARG0 – "The lawyers" to the predicate "went", is represented with the string NP↑S↓VP↓VBD.

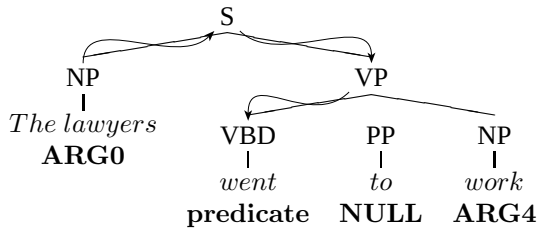

Figure 1. Illustration of path NP↑S↓VP↓VBD

Phrase Type – The syntactic category (NP, PP, S, etc.) of the phrase corresponding to the semantic role.

Position – Whether the phrase is before or after the governing predicate.

Voice – Whether the governing predicate is realized as an active or passive construction.

Head Word – The syntactic head of the phrase.

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 "went" is VP→VBD-PP-NP.

We use SVM as the classifier in the ONE *vs* ALL formalism, where an SVM is trained for each class (ARG0-5, ARGMs, and NULL) to discriminate between that class and all others. We found it efficient to divide the classification process into three stages:

1. A binary NULL *vs* NON-NULL classifier labels each node as NULL or as being some argument class. A threshold is set so that nodes with very high confidence of being NULL are pruned.

2. In a second pass, each node not pruned in the first stage is classified as one of the set of argument classes or as NULL using the ONE *vs* ALL strategy. In this collection of binary classifiers, the NULL *vs* NON-NULL classifier is trained on nodes that weren't pruned by the first pass, and so is different than the one in the first pass.

3. Overlapping argument assignments are disallowed. Since there are no overlapping roles in the training set, this is another constraint that can be enforced and results in a significant increase in precision with little or no reduction in recall.

4. Experimental Configurations and Results

For our experiments, we used tinySVM[3] along with YamCha[4] as the SVM training and test software. The system was optimized on three parameters: a) The kernel function used - polynomial with degree 2; b) The cost per unit violation of the margin (C=1), and c) Tolerance of the termination criterion (e=0.001).

As a baseline, we trained a system using the set of features listed earlier. The Precision, Recall and F_1 measure are shown in the Baseline row of Table 1. We also tested four new features, Verb Clusters, Named Entities, Partial-Path and Head Word Part-of-Speech[5].

Verb clustering – In order to improve performance on verbs that are unseen in the training set, we clustered verbs into 64 classes using the probabilistic co-occurrence model of Hofmann [10] and using a distance function derived from Lin's database of verb-direct object relations [12]. The verb class of the current predicate was added as a feature. The performance improvement is shown in Table 1.

	No Overlaps		
	P	R	F_1
Baseline	85	79	82
With verb clusters	86	81	84

Table 1. Improvement on adding verb-cluster

Named Entities in the constituents – Another obvious improvement was considering the presence of named entities present in the constituents. We tagged 7 named entities (PERSON, ORGANIZATION, LOCATION, PERCENT, MONEY, TIME, DATE) using Identifinder [2] and added them as binary features. This feature is true if the entity is contained in the constituent. On the task of assigning labels to constituents known to represent either "core" or "adjunctive" arguments, adding this feature increased the accuracy from 87.74% to 88.24%. The most significant improvement was for adjunct roles like temporals (ARGM-TMP) and locatives (ARGM-LOC) as shown in Table 2. Named Entities are used in the argument classifiers but not in the NULL *vs* NON-NULL classifier. We found the classifier degraded when this feature was added.

	ARGM-LOC			ARGM-TMP		
	P	R	F_1	P	R	F_1
Without NE	64	55	59	80	85	82
With NE	71	67	69	84	87	85

Table 2. Improvement using Named Entities

Partial Path – We tried generalizing the Path feature by setting its value to the part of the original Path feature that goes from the constituent to the common parent. Partial

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

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

[5] Unless mentioned otherwise, the test set comprises perfect hand-corrected, "gold-standard" parses.

Path for the Path illustrated in Figure 1 is NP↑S. On the task of assigning labels to constituents known to represent "core arguments", adding this feature increased the accuracy from 93.7% to 94%.

Head Word Part-of-Speech (POS) – Adding the Head Word POS improved NULL *vs* NON-NULL classification accuracy from 91% to 92%

4.1. Alternative Pruning Strategies. A preliminary error analysis suggested that the biggest confusion was between NULL and NON-NULL roles, therefore we decided to re-examine our strategy of filtering out nodes that have a high likelihood of being NULL in a first pass. To do this, we converted the raw SVM scores to probabilities by fitting a sigmoid function [14]. We trained and tested systems for three conditions:

1. System I, a one pass system where all ONE *vs* ALL classifiers are trained on all the data. This has considerably higher training time as compared to the other two.
2. System II, which uses a NULL *vs* NON-NULL classifier in a first pass. The difference is that here, all nodes labeled NULL are filtered (not just high confidence ones.)
3. System III, which also uses a NULL *vs* NON-NULL classifier, but filters out nodes with high confidence of being NULL in a first pass.

A detailed description of all possible formulations, are described at length in [9, 8]

Table 3 shows performance on the task of identifying and labeling PropBank arguments. The two-stage System III performs the best. It slightly outperforms the system trained on all the data. The fact that about 80% of the nodes in a tree are NULL, and we have to train only one classifier on the entire data, there is a considerable saving in training time. Therefore, we decided to continue using this strategy.

	No Overlaps		
	P	R	F_1
SVM System I	87	80	83.3
SVM System II	84	80	81.9
SVM System III	86	81	83.4

Table 3. Comparing Pruning Strategies

5. Comparing Performance with Other Systems

We evaluated our system in a number of ways. First, we compare it against 4 other shallow parsers in the literature. Second, we compare the performance of our system when using gold-standard parses versus when using an more realistic parser – Charniak Parser [3]. Finally, we compare the performance of our parser on "core arguments" (ARG0-ARG5).

In comparing systems, results are reported for tree types of tasks:

1. *Argument Identification* - Given a (correct) parse tree, label each node as NULL or as being some argument (the NULL - NON-NULL discrimination).
2. *Argument Classification* - Given the (correct) set of nodes in the tree that are arguments, label each node with the argument class label.
3. *Combined Identification and Classification* - This is the real usage scenario where the system must classify nodes as NULL or some specific argument.

5.1. Description of the Systems.

5.1.1 The Gildea and Palmer (G&P) System. This system uses the same features used by G&J [6], which are the ones that we started with. They report results on the December 2001 release of PropBank.

5.1.2 The Surdeanu *et al.* System. Surdeanu *et al.* [16] report results on two systems. One that uses exactly the same features as the G&J [6] system. We call this "Surdeanu System I." [16] They then show improved performance of another system – "Surdeanu System II," [16] which uses some additional features. They use the July 2002 release of PropBank.

5.1.3 The Gildea and Hockenmaier (G&H) System. The G&H [5] system uses features extracted from Combinatory Categorial Grammar (CCG) corresponding to the features that were used by G&J [6] and G&P [7] systems. They use a slightly newer – November 2002 release of PropBank. We will refer to this as "G&H System I". They also report performance on the Treebank-based data – "G&H System II."

5.1.4 The Chen and Rambow (C&R) System. Chen and Rambow report on two different systems. The first "C&R System I" uses surface syntactic features much like the G&P [7] system. The second "C&R System II" uses additional syntactic and semantic representations that are extracted from a Tree Adjoining Grammar (TAG).

5.2. Role Classification Using Known Boundaries. Table 4 compares the role 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.

"C&R System II" [4] uses some additional syntactic features extracted from TAG. The "C&R System I" [4] that uses the almost the same features as the SVM System performs considerably worse. The test sets of the "C&R System I" [4] and SVM System are not identical, but nevertheless, the rough comparison is valuable.

5.3. Argument Identification (NULL vs NON-NULL). Table 5 compares the results of the task of identifying the parse constituents that represent semantic arguments. As expected, the performance degrades considerably when we

Classes	*System*	*Gold*	*Automatic*
		Accuracy	Accuracy
ARG0-5	SVM	88	87
+	G&P	77	74
ARGMS	Surdeanu System II	84	-
	Surdeanu System I	79	-
CORE ARGUMENTS	SVM	93.9	90
(ARG0-5)	C&R System II	93.5	-
	C&R System I	92.4	-

Table 4. Argument classification

extract features from an automatic parse as opposed to a gold-standard parse. This indicates that the syntactic parser performance directly influences the role 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*	*Gold*			*Automatic*		
		P	R	F_1	P	R	F_1
ARG0-5	SVM	94	90	92	89	80	84
+	Surdeanu System II	-	-	89	-	-	-
ARGMS	Surdeanu System I	85	84	85	-	-	-

Table 5. Argument identification

5.4. Argument Identification and Tagging. Table 6 shows the results for the task where the system first identifies candidate argument boundaries and then labels them with the most likely role as discussed in Sections 3 and 4. 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*	*Gold*			*Automatic*		
		P	R	F_1	P	R	F_1
ARG0-5	SVM System III	86	81	83	82	73	77
+	G&H System I	76	68	72	71	63	67
ARGMS	G&H System II	79	70	74	73	61	66
	G&P	71	64	67	58	50	54
ARG0-5	SVM System III	89	85	87	85	77	81
	G&H System I	82	79	80	76	73	75
	G&H System II	85	82	84	76	70	73
	C&R System II	-	-	-	65	75	70

Table 6. Identification and classification

6. Conclusions

We have extended the work of Gildea and Jurafsky [6] on shallow semantic role labeling. We first made a number of small augmentations to their system which generalizes the statistical power of the original algorithm, improving the precision and recall significantly on test data. We then replaced the probability estimators of the original system with an SVM. This resulted in a substantial improvement the system's overall performance. A detailed comparison of our results with those reported by other groups working on similar tasks indicate that ours outperforms all. One drawback of the current system is that it labels each argument in a sentence independent of the others. We plan to overcome this by converting the SVM output into an n-best lattice and using other features such as role language model score.

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 parser; Martha Palmer for providing us with the PropBank data; Valerie Krugler and Karin Kipper for mapping the PropBank arguments to thematic roles.

References

[1] C. F. Baker, C. J. Fillmore, and J. B. Lowe. The Berkeley FrameNet project. In *COLING/ACL-98*, Montreal, 1998. ACL.

[2] D. M. Bikel, R. Schwartz, and R. M. Weischedel. An algorithm that learns what's in a name. *Machine Learning*, 34:211–231, 1999.

[3] E. Chaniak. Immediate-head parsing for language models. In *Proceedings of the 39th ACL*, Toulouse, France, 2001.

[4] J. Chen and O. Rambow. Use of deep linguistics features for the recognition and labeling of semantic arguments. In *Proceedings of the EMNLP*, Sapporo, Japan, 2003.

[5] D. Gildea and J. Hockenmaier. Identifying semantic roles using combinatory categorial grammar. In *Proceedings of the EMNLP*, Sapporo, Japan, 2003.

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

[7] D. Gildea and M. Palmer. The necessity of syntactic parsing for predicate argument recognition. In *Proceedings of the 40th ACL*, Philadelphia, PA, 2002.

[8] K. Hacioglu, S. Pradhan, W. Ward, J. Martin, and D. Jurafsky. Shallow semantic parsing using support vector machines. Technical Report TR-CSLR-2003-1, Center for Spoken Language Research, Boulder, Colorado, 2003.

[9] K. Hacioglu and W. Ward. Target word detection and semantic role chunking using support vector machines. In *Proceedings of the Human Language Technology Conference*, Edmonton, Canada, 2003.

[10] T. Hofmann and J. Puzicha. Statistical models for co-occurrence data. Memo, Massachussetts Institute of Technology Artificial Intelligence Laboratory, Feb. 1998.

[11] P. Kingsbury and M. Palmer. From Treebank to PropBank. In *Proceedings of the LREC*, Las Palmas, Canary Islands, Spain, 2002.

[12] D. Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the COLING-ACL*, Montreal, Canada, 1998.

[13] M. Marcus, G. Kim, M. A. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. The penn treebank: Annotating predicate argument structure, 1994.

[14] J. 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.

[15] G. N. Sean Wallis. Knowledge discovery in grammatically analysed corpora. *Data Mining and Knowledge Discovery*, 5(4):305–335, 2001.

[16] M. Surdeanu, S. Harabagiu, J. Williams, and P. Aarseth. Using predicate-argument structures for information extraction. In *Proceedings of the ACL*, Sapporo, Japan, 2003.

[17] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.