Semantic Role Labeling Using Different Syntactic Views*

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

Daniel Jurafsky
Department of Linguistics,
Stanford University,
Stanford, CA 94305
jurafsky@stanford.edu

Abstract

Semantic role labeling is the process of annotating the predicate-argument structure in text with semantic labels. In this paper we present a state-of-the-art baseline semantic role labeling system based on Support Vector Machine classifiers. We show improvements on this system by: i) adding new features including features extracted from dependency parses, ii) performing feature selection and calibration and iii) combining parses obtained from semantic parsers trained using different syntactic views. Error analysis of the baseline system showed that approximately half of the argument identification errors resulted from parse errors in which there was no syntactic constituent that aligned with the correct argument. In order to address this problem, we combined semantic parses from a Minipar syntactic parse and from a chunked syntactic representation with our original baseline system which was based on Charniak parses. All of the reported techniques resulted in performance improvements.

1 Introduction

Semantic Role Labeling is the process of annotating the predicate-argument structure in text with semantic labels (Gildea and Jurafsky, 2000; Gildea and Jurafsky, 2002; Gildea and Palmer, 2002; Surdeanu et al., 2003; Hacioglu and Ward, 2003; Chen and Rambow, 2003; Gildea and Hockenmaier, 2003; Pradhan et al., 2004; Hacioglu, 2004). The architecture underlying all of these systems introduces two distinct sub-problems: the identification of syntactic constituents that are semantic roles for a given predicate, and the labeling of the those constituents with the correct semantic role.

A detailed error analysis of our baseline system indicates that the identification problem poses a significant bottleneck to improving overall system performance. The baseline system's accuracy on the task of labeling nodes known to represent semantic arguments is 90%. On the other hand, the system's performance on the identification task is quite a bit lower, achieving only 80% recall with 86% precision. There are two sources of these identification errors: i) failures by the system to identify all and only those constituents that correspond to semantic roles, *when those constituents are present in the syntactic analysis*, and ii) failures by the syntactic analyzer to provide the constituents that align with correct arguments. The work we present here is tailored to address these two sources of error in the identification problem.

The remainder of this paper is organized as follows. We first describe a baseline system based on the best published techniques. We then report on two sets of experiments using techniques that improve performance on the problem of finding arguments when they are present in the syntactic analysis. In the first set of experiments we explore new

*This research was partially supported by the ARDA AQUAINT program via contract OCG4423B and by the NSF via grants IS-9978025 and ITR/HCI 0086132

Proceedings of the 43rd Annual Meeting of the ACL, pages 581–588,
Ann Arbor, June 2005.

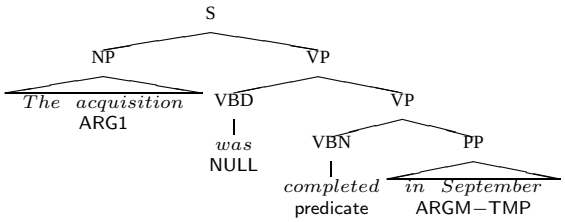
features, including features extracted from a parser that provides a different syntactic view – a Combinatory Categorial Grammar (CCG) parser (Hockenmaier and Steedman, 2002). In the second set of experiments, we explore approaches to identify optimal subsets of features for each argument class, and to calibrate the classifier probabilities.

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

2 Baseline System

For our experiments, we use Feb 2004 release of PropBank[1] (Kingsbury and Palmer, 2002; Palmer et al., 2005), a corpus in which predicate argument relations are marked for verbs in the Wall Street Journal (WSJ) part of the Penn TreeBank (Marcus et al., 1994). PropBank was constructed by assigning semantic arguments to constituents of hand-corrected TreeBank parses. Arguments of a verb are labeled ARG0 to ARG5, where ARG0 is the PROTO-AGENT, ARG1 is the PROTO-PATIENT, etc. In addition to these CORE ARGUMENTS, additional ADJUNCTIVE ARGUMENTS, referred to as ARGMs are also marked. Some examples are ARGM-LOC, for locatives; ARGM-TMP, for temporals; ARGM-MNR, for manner, etc. Figure 1 shows a syntax tree along with the argument labels for an example extracted from PropBank. We use Sections 02-21 for training, Section 00 for development and Section 23 for testing.

We formulate the semantic labeling problem as a multi-class classification problem using Support Vector Machine (SVM) classifier (Hacioglu et al., 2003; Pradhan et al., 2003; Pradhan et al., 2004) TinySVM[2] along with YamCha[3] (Kudo and Matsumoto, 2000; Kudo and Matsumoto, 2001) are used to implement the system. Using what is known as the ONE VS ALL classification strategy, n binary classifiers are trained, where n is number of semantic classes including a NULL class.



[ARG1 The acquisition] was [predicate completed] [ARGM−TMP in September].

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

The baseline feature set is a combination of features introduced by Gildea and Jurafsky (2002) and ones proposed in Pradhan et al., (2004), Surdeanu et al., (2003) and the *syntactic-frame* feature proposed in (Xue and Palmer, 2004). Table 1 lists the features used.

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

Table 1: Features used in the Baseline system

As described in (Pradhan et al., 2004), we post-process the n-best hypotheses using a trigram language model of the argument sequence.

We analyze the performance on three tasks:

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

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

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

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

- **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.

ALL ARGS	Task	P (%)	R (%)	F_1	A (%)
HAND	Id.	96.2	95.8	96.0	
	Classification	-	-	-	93.0
	Id. + Classification	89.9	89.0	89.4	
AUTOMATIC	Id.	86.8	80.0	83.3	
	Classification	-	-	-	90.1
	Id. + Classification	80.9	76.8	78.8	

Table 2: Baseline system performance on all tasks using hand-corrected parses and automatic parses on PropBank data.

Table 2 shows the performance of the system using the hand corrected, TreeBank parses (HAND) and using parses produced by a Charniak parser (AUTOMATIC). Precision (P), Recall (R) and F_1 scores are given for the identification and combined tasks, and Classification Accuracy (A) for the classification task.

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

Motivated by this severe degradation in argument identification performance for automatic parses, we examined a number of techniques for improving argument identification. We made a number of changes to the system which resulted in improved performance. The changes fell into three categories: i) new features, ii) feature selection and calibration, and iii) combining parses from different syntactic representations.

3 Additional Features

3.1 CCG Parse Features

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

- **Phrase type –** This is the category of the maximal projection between the two words – the predicate and the dependent word.
- **Categorial Path –** This is a feature formed by concatenating the following three values: i) category to which the dependent word belongs, ii) the direction of dependence and iii) the slot in the category filled by the dependent word.
- **Tree Path –** This is the categorial analogue of the path feature in the Charniak parse based system, which traces the path from the dependent word to the predicate through the binary CCG tree.

Parallel to the hand-corrected TreeBank parses, we also had access to correct CCG parses derived from the TreeBank (Hockenmaier and Steedman, 2002a). We performed two sets of experiments. One using the correct CCG parses, and the other using parses obtained using StatCCG[4] parser (Hockenmaier and Steedman, 2002). We incorporated these features in the systems based on hand-corrected TreeBank parses and Charniak parses respectively. For each constituent in the Charniak parse tree, if there was a dependency between the head word of the constituent and the predicate, then the corresponding CCG features for those words were added to the features for that constituent. Table 3 shows the performance of the system when these features were added. The corresponding baseline performances are mentioned in parentheses.

3.2 Other Features

We added several other features to the system. Position of the clause node (S, SBAR) seems to be

[4]Many thanks to Julia Hockenmaier for providing us with the CCG bank as well as the StatCCG parser.

ALL ARGS	Task	P (%)	R (%)	F_1
HAND	Id.	97.5 (96.2)	96.1 (95.8)	96.8 (96.0)
	Id. + Class.	91.8 (89.9)	90.5 (89.0)	91.2 (89.4)
AUTOMATIC	Id.	87.1 (86.8)	80.7 (80.0)	83.8 (83.3)
	Id. + Class.	81.5 (80.9)	77.2 (76.8)	79.3 (78.8)

Table 3: Performance improvement upon adding CCG features to the Baseline system.

an important feature in argument identification (Hacioglu et al., 2004) therefore we experimented with four clause-based path feature variations. We added the predicate context to capture predicate sense variations. For some adjunctive arguments, punctuation plays an important role, so we added some punctuation features. All the new features are shown in Table 4

CLAUSE-BASED PATH VARIATIONS: I. Replacing all the nodes in a path other than clause nodes with an "*". For example, the path NP↑S↑VP↑SBAR↑NP↑VP↓VBD becomes NP↑S↑*S↑*↑*↓VBD II. Retaining only the clause nodes in the path, which for the above example would produce NP↑S↑S↓VBD, III. Adding a binary feature that indicates whether the constituent is in the same clause as the predicate, IV. collapsing the nodes between S nodes which gives NP↑S↑NP↑VP↓VBD.
PATH N-GRAMS: This feature decomposes a path into a series of trigrams. For example, the path NP↑S↑VP↑SBAR↑NP↑VP↓VBD becomes: NP↑S↑VP, S↑VP↑SBAR, VP↑SBAR↑NP, SBAR↑NP↑VP, etc. We used the first ten trigrams as ten features. Shorter paths were padded with nulls.
SINGLE CHARACTER PHRASE TAGS: Each phrase category is clustered to a category defined by the first character of the phrase label.
PREDICATE CONTEXT: Two words and two word POS around the predicate and including the predicate were added as ten new features.
PUNCTUATION: Punctuation before and after the constituent were added as two new features.
FEATURE CONTEXT: Features for argument bearing constituents were added as features to the constituent being classified.

Table 4: Other Features

4 Feature Selection and Calibration

In the baseline system, we used the same set of features for all the n binary ONE VS ALL classifiers. Error analysis showed that some features specifically suited for one argument class, for example, core arguments, tend to hurt performance on some adjunctive arguments. Therefore, we thought that selecting subsets of features for each argument class might improve performance. To achieve this, we performed a simple feature selection procedure. For each argument, we started with the set of features introduced by (Gildea and Jurafsky, 2002). We pruned this set by training classifiers after leaving out one feature at a time and checking its performance on a development set. We used the χ^2 significance while making pruning decisions. Following that, we added each of the other features one at a time to the pruned baseline set of features and selected ones that showed significantly improved performance. Since the feature selection experiments were computationally intensive, we performed them using 10k training examples.

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

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

Table 5: Performance improvement on selecting features per argument and calibrating the probabilities on 10k training data.

Foster and Stine (2004) show that the pool-adjacent-violators (PAV) algorithm (Barlow et al., 1972) provides a better method for converting raw classifier scores to probabilities when Platt's algorithm fails. The probabilities resulting from either conversions may not be properly calibrated. So, we binned the probabilities and trained a warping function to calibrate them. For each argument classifier, we used both the methods for converting raw SVM scores into probabilities and calibrated them using a development set. Then, we visually inspected the calibrated plots for each classifier and chose the method that showed better calibration as the calibration procedure for that classifier. Plots of the predicted probabilities versus true probabilities for the ARGM-TMP VS ALL classifier, before and after calibration are shown in Figure 2. The performance improvement over a classifier that is trained using all the features for all the classes is shown in Table 5.

Table 6 shows the performance of the system after adding the CCG features, additional features ex-

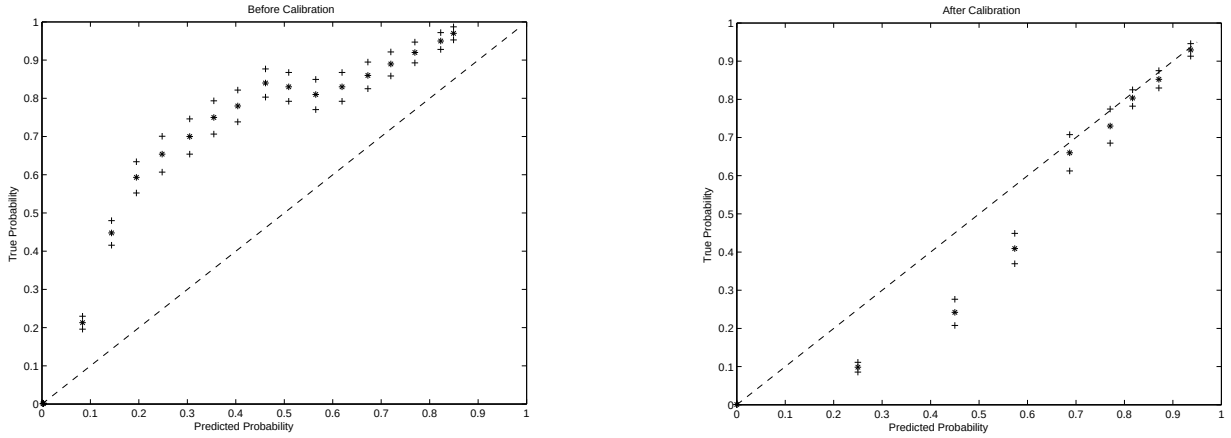


Figure 2: Plots showing true probabilities versus predicted probabilities before and after calibration on the test set for ARGM-TMP.

tracted from the Charniak parse tree, and performing feature selection and calibration. Numbers in parentheses are the corresponding baseline performances.

TASK	P (%)	R (%)	F_1	A (%)
Id.	86.9 (86.8)	84.2 (80.0)	85.5 (83.3)	92.0 (90.1)
Class.	-	-	-	
Id. + Class.	82.1 (80.9)	77.9 (76.8)	79.9 (78.8)	

Table 6: Best system performance on all tasks using automatically generated syntactic parses.

5 Alternative Syntactic Views

Adding new features can improve performance when the syntactic representation being used for classification contains the correct constituents. Additional features can't recover from the situation where the parse tree being used for classification doesn't contain the correct constituent representing an argument. Such parse errors account for about 7% absolute of the errors (or, about half of 12.7%) for the Charniak parse based system. To address these errors, we added two additional parse representations: i) Minipar dependency parser, and ii) chunking parser (Hacioglu et al., 2004). The hope is that these parsers will produce different errors than the Charniak parser since they represent different syntactic views. The Charniak parser is trained on the Penn TreeBank corpus. Minipar is a rule based dependency parser. The chunking parser is trained on PropBank and produces a flat syntactic representation that is very different from the full parse tree produced by Charniak. A combination of the three different parses could produce better results than any single one.

5.1 Minipar-based Semantic Labeler

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

The set of words under each node in Minipar's dependency tree form a contiguous segment in the original sentence and correspond to the constituent in a constituent tree. We formulate the semantic labeling problem in the same way as in a constituent structure parse, except we classify the nodes that represent head words of constituents. A similar formulation using dependency trees derived from TreeBank was reported in Hacioglu (Hacioglu, 2004). In that experiment, the dependency trees were derived from hand-corrected TreeBank trees using head word rules. Here, an SVM is trained to assign PropBank argument labels to nodes in Minipar dependency trees using the following features:

Table 8 shows the performance of the Minipar-based semantic parser.

Minipar performance on the PropBank corpus is substantially worse than the Charniak based system. This is understandable from the fact that Minipar is not designed to produce constituents that would exactly match the constituent segmentation used in TreeBank. In the test set, about 37% of the argu-

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

Table 7: Features used in the Baseline system using Minipar parses.

Task	P (%)	R (%)	F_1
Id.	73.5	43.8	54.6
Id. + Classification	66.2	36.7	47.2

Table 8: Baseline system performance on all tasks using Minipar parses.

ments do not have corresponding constituents that match its boundaries. In experiments reported by Hacioglu (Hacioglu, 2004), a mismatch of about 8% was introduced in the transformation from hand-corrected constituent trees to dependency trees. Using an errorful automatically generated tree, a still higher mismatch would be expected. In case of the CCG parses, as reported by Gildea and Hockenmaier (2003), the mismatch was about 23%. A more realistic way to score the performance is to score tags assigned to head words of constituents, rather than considering the exact boundaries of the constituents as reported by Gildea and Hockenmaier (2003). The results for this system are shown in Table 9.

	Task	P (%)	R (%)	F_1
CHARNIAK	Id.	92.2	87.5	89.8
	Id. + Classification	85.9	81.6	83.7
MINIPAR	Id.	83.3	61.1	70.5
	Id. + Classification	72.9	53.5	61.7

Table 9: Head-word based performance using Charniak and Minipar parses.

5.2 Chunk-based Semantic Labeler

Hacioglu has previously described a chunk based semantic labeling method (Hacioglu et al., 2004). This system uses SVM classifiers to first chunk input text into flat chunks or base phrases, each labeled with a syntactic tag. A second SVM is trained to assign semantic labels to the chunks. The system is trained on the PropBank training data.

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

Table 10: Features used by chunk based classifier.

Table 10 lists the features used by this classifier. For each token (base phrase) to be tagged, a set of features is created from a fixed size context that surrounds each token. In addition to the above features, it also uses previous semantic tags that have already been assigned to the tokens contained in the linguistic context. A 5-token sliding window is used for the context.

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

Table 11: Semantic chunker performance on the combined task of Id. and classification.

SVMs were trained for begin (B) and inside (I) classes of all arguments and outside (O) class for a total of 78 one-vs-all classifiers. Again, TinySVM[5] along with YamCha[6] (Kudo and Matsumoto, 2000; Kudo and Matsumoto, 2001) are used as the SVM training and test software.

Table 11 presents the system performances on the PropBank test set for the chunk-based system.

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

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

6 Combining Semantic Labelers

We combined the semantic parses as follows: i) scores for arguments were converted to calibrated probabilities, and arguments with scores below a threshold value were deleted. Separate thresholds were used for each parser. ii) For the remaining arguments, the more probable ones among overlapping ones were selected. In the chunked system, an argument could consist of a sequence of chunks. The probability assigned to the begin tag of an argument was used as the probability of the sequence of chunks forming an argument. Table 12 shows the performance improvement after the combination. Again, numbers in parentheses are respective baseline performances.

TASK	P (%)	R (%)	F_1
Id.	85.9 (86.8)	88.3 (80.0)	87.1 (83.3)
Id. + Class.	81.3 (80.9)	80.7 (76.8)	81.0 (78.8)

Table 12: Constituent-based best system performance on argument identification and argument identification and classification tasks after combining all three semantic parses.

The main contribution of combining both the Minipar based and the Charniak-based parsers was significantly improved performance on ARG1 in addition to slight improvements to some other arguments. Table 13 shows the effect on selected arguments on sentences that were altered during the the combination of Charniak-based and Chunk-based parses.

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

	Before			After		
	P (%)	R (%)	F_1	P (%)	R (%)	F_1
Overall	94.8	53.4	68.3	80.9	73.8	**77.2**
ARG0	96.0	85.7	90.5	92.5	89.2	**90.9**
ARG1	71.4	13.5	22.7	59.4	59.4	**59.4**
ARG2	100.0	20.0	33.3	50.0	20.0	28.5
ARGM-DIS	100.0	40.0	57.1	100.0	100.0	**100.0**

Table 13: Performance improvement on parses changed during pair-wise Charniak and Chunk combination.

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

To give an idea of what the potential improvements of the combinations could be, we performed an oracle experiment for a combined system that tags head words instead of exact constituents as we did in case of Minipar-based and Charniak-based semantic parser earlier. In case of chunks, first word in prepositional base phrases was selected as the head word, and for all other chunks, the last word was selected to be the head word. If the correct argument was found present in either the Charniak, Minipar or Chunk hypotheses then that was selected. The results for this are shown in Table 14. It can be seen that the head word based performance almost approaches the constituent based performance reported on the hand-corrected parses in Table 3 and there seems to be considerable scope for improvement.

	Task	P (%)	R (%)	F_1
C	Id.	92.2	87.5	89.8
	Id. + Classification	85.9	81.6	83.7
C+M	Id.	98.4	90.6	94.3
	Id. + Classification	93.1	86.0	89.4
C+CH	Id.	98.9	88.8	93.6
	Id. + Classification	92.5	83.3	87.7
C+M+CH	Id.	99.2	92.5	95.7
	Id. + Classification	94.6	88.4	91.5

Table 14: Performance improvement on head word based scoring after oracle combination. Charniak (C), Minipar (M) and Chunker (CH).

Table 15 shows the performance improvement in the actual system for pairwise combination of the parsers and one using all three.

	Task	P (%)	R (%)	F_1
C	Id.	92.2	87.5	89.8
	Id. + Classification	85.9	81.6	83.7
C+M	Id.	91.7	89.9	90.8
	Id. + Classification	85.0	83.9	84.5
C+CH	Id.	91.5	91.1	91.3
	Id. + Classification	84.9	84.3	84.7
C+M+CH	Id.	91.5	91.9	91.7
	Id. + Classification	85.1	85.5	85.2

Table 15: Performance improvement on head word based scoring after combination. Charniak (C), Minipar (M) and Chunker (CH).

7 Conclusions

We described a state-of-the-art baseline semantic role labeling system based on Support Vector Machine classifiers. Experiments were conducted to evaluate three types of improvements to the system: i) adding new features including features extracted from a Combinatory Categorial Grammar parse, ii) performing feature selection and calibration and iii) combining parses obtained from semantic parsers trained using different syntactic views. We combined semantic parses from a Minipar syntactic parse and from a chunked syntactic representation with our original baseline system which was based on Charniak parses. The belief was that semantic parses based on different syntactic views would make different errors and that the combination would be complimentary. A simple combination of these representations did lead to improved performance.

8 Acknowledgements

This research was partially supported by the ARDA AQUAINT program via contract OCG4423B and by the NSF via grants IS-9978025 and ITR/HCI 0086132. Computer time was provided by NSF ARI Grant #CDA-9601817, NSF MRI Grant #CNS-0420873, NASA AIST grant #NAG2-1646, DOE SciDAC grant #DE-FG02-04ER63870, NSF sponsorship of the National Center for Atmospheric Research, and a grant from the IBM Shared University Research (SUR) program.

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; Dan Gildea and Julia Hockenmaier for providing the gold standard CCG parser information, and all the anonymous reviewers for their helpful comments.

References

R. E. Barlow, D. J. Bartholomew, J. M. Bremmer, and H. D. Brunk. 1972. *Statistical Inference under Order Restrictions*. Wiley, New York.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL*, pages 132–139, Seattle, Washington.

John Chen and Owen Rambow. 2003. Use of deep linguistics features for the recognition and labeling of semantic arguments. In *Proceedings of the EMNLP*, Sapporo, Japan.

Dean P. Foster and Robert A. Stine. 2004. Variable selection in data mining: building a predictive model for bankruptcy. *Journal of American Statistical Association*, 99, pages 303–313.

Dan Gildea and Julia Hockenmaier. 2003. Identifying semantic roles using combinatory categorial grammar. In *Proceedings of the EMNLP*, Sapporo, Japan.

Daniel Gildea and Daniel Jurafsky. 2000. Automatic labeling of semantic roles. In *Proceedings of ACL*, pages 512–520, Hong Kong, October.

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

Daniel Gildea and Martha Palmer. 2002. The necessity of syntactic parsing for predicate argument recognition. In *Proceedings of ACL*, Philadelphia, PA.

Kadri Hacioglu. 2004. Semantic role labeling using dependency trees. In *Proceedings of COLING*, Geneva, Switzerland.

Kadri Hacioglu and Wayne Ward. 2003. Target word detection and semantic role chunking using support vector machines. In *Proceedings of HLT/NAACL*, Edmonton, Canada.

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.

Kadri Hacioglu, Sameer Pradhan, Wayne Ward, James Martin, and Daniel Jurafsky. 2004. Semantic role labeling by tagging syntactic chunks. In *Proceedings of CoNLL-2004, Shared Task – Semantic Role Labeling*.

Kadri Hacioglu. 2004a. A lightweight semantic chunking model based on tagging. In *Proceedings of HLT/NAACL*, Boston, MA.

Julia Hockenmaier and Mark Steedman. 2002. Generative models for statistical parsing with combinatory grammars. In *Proceedings of the ACL*, pages 335–342.

Julia Hockenmaier and Mark Steedman. 2002a. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002)*, Las Palmas, Canary Islands, Spain.

Paul Kingsbury and Martha Palmer. 2002. From Treebank to PropBank. In *Proceedings of LREC*, Las Palmas, Canary Islands, Spain.

Taku Kudo and Yuji Matsumoto. 2000. Use of support vector learning for chunk identification. In *Proceedings of CoNLL and LLL*, pages 142–144.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of the NAACL*.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):343–360.

Dekang Lin. 1998. Dependency-based evaluation of MINIPAR. In *In Workshop on the Evaluation of Parsing Systems*, Granada, Spain.

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.

Martha Palmer, Dan Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. To appear *Computational Linguistics*.

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

Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James Martin, and Dan Jurafsky. 2003. Semantic role parsing: Adding semantic structure to unstructured text. In *Proceedings of ICDM*, Melbourne, Florida.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Dan Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of HLT/NAACL*, Boston, MA.

Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of ACL*, Sapporo, Japan.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP*, Barcelona, Spain.